UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

TRABAJO FIN DE GRADO

Diseño de videojuegos para el aprendizaje

Javier Rubio Mingorance
Tutor: Sacha Gómez Moñivas
Ponente: Simone Santini

Septiembre 2014

1. RESUMEN DEL TFG

El objetivo principal que se ha perseguido a la hora de realizar este trabajo ha sido el de permitir diseñar videojuegos educativos de una manera sencilla a usuarios con escasos conocimientos informáticos.

Durante el desarrollo del trabajo se han generado dos aplicaciones diferentes: se ha desarrollado por un lado un motor de videojuegos configurable y por otra parte una aplicación que permite al usuario diseñar y crear sus propios videojuegos utilizando el motor creado anteriormente.

El motor configurable de videojuegos se ha implementado sobre el motor XNA 4.0 desarrollado por la empresa Microsoft, dicho motor utiliza el lenguaje de programación C#. Las principales razones que me han llevado a utilizar XNA 4.0 han sido las siguientes:

- En primer lugar XNA 4.0 es fácilmente accesible para una gran cantidad de usuarios, es gratuito y no se requiere un sistema potente para ejecutar los juegos creados con este motor.

- Por otra parte, ya tenía una cierta experiencia tanto con el motor XNA 4.0 como con el lenguaje C#, concretamente el motor de XNA lo utilice cuando curse la asignatura “Introducción a la programación de videojuegos y gráficos” y el lenguaje C# lo había utilizado para desarrollar algunas aplicaciones.

Respecto a la aplicación encargada de la creación de videojuegos, he optado por utilizar el lenguaje Visual Basic 6. He decidido utilizar este lenguaje principalmente porque tengo experiencia a la hora de desarrollar aplicaciones escritas en este lenguaje.

En el desarrollo del proyecto he intentado priorizar la sencillez de uso del producto para que pueda ser utilizado por usuarios con un escaso nivel de conocimientos informáticos, ya que este era uno de los objetivos principales del TFG.

Por ello todo lo referente al código del juego es invisible al diseñador del videojuego, de manera que este solo tiene que preocuparse del diseño. Es decir el usuario solo tiene que preocuparse por cómo quiere que sea su juego y no en cómo implementarlo.

El desarrollo de este proyecto se ha estructurado en tres fases claramente diferenciadas entre sí:

- Primera fase – Esta fase ha sido la fase principal de desarrollo de este proyecto, en ella se han aclarado las principales líneas a seguir durante el desarrollo de la aplicación y se ha generado una primera versión de las aplicaciones.

- Segunda fase – En esta fase se ha interaccionado con personas del departamento de Historia de la Universidad Autónoma de Madrid para obtener su opinión de la aplicación y que cosas creen que deberían de ser cambiadas para hacer la aplicación más fácil de utilizar y más completa.

- Tercera fase – Durante esta última fase del proyecto se ha implementado diversos cambios a partir de las opiniones obtenidas anteriormente para generar la versión final de la aplicación.

Mediante este proceso de desarrollo en tres fases se ha buscado conseguir una aplicación que no sea difícil de utilizar por usuarios sin conocimientos informáticos avanzados pero que a su vez tenga una funcionalidad lo más completa posible.

1. SUMMARY

The main goal of this project was to allow people without advanced knowledge of computers to design their own educational videogames.

This project can be divided in two different applications: A configurable videogame engine and an application that allows the user to create and design their own videogames using that engine.

The configurable videogame engine has been developed using the XNA 4.0 engine, this engine uses C# programming language. I decided to use this engine mainly because:

- First of all, XNA 4.0 is easily accessible for a large amount of users, it is free and it doesn't require a powerful computer to run the games.

- I had some previous experience using XNA and the programming language C#, specifically I used XNA when I took part on the subject called "Introducción a la programación de videojuegos y gráficos" and I have also used the language C# to develop some applications.

In reference to the application that allows the user to create and design their own videogames, I decided to use the programming language Visual Basic 6, I decided to use this language because I got some experience with it because I developed a few applications in this language.

During the development of this project, I tried to focus into making an easy to use application so it can be used by people without advanced knowledge of computers, since this was one of the main goals of the project.

Because of this, all the coding part of the project is invisible to the videogame designer, this way the designer only has to worry about the design, in other words he only needs to think about how he wants the game to be and not about how to make it work.

The development of this project has been structured in three different phases:

- First phase – This phase has been the main phase of the development of the project, during this phase we have set the main lanes to follow during the process of development and the first versions of the applications were developed.

- Second phase – During this phase we worked with people of the History department of the University in order to get their opinion about the application and what would they change in the application to make it better and easier to use.

- Third phase – During this last phase of the project, we made changes in the application based on the feedback received from the people of the History department, we make this changes in order to make the final version of the application as good as possible.

What we tried to achieve with this three phases process was not only to get and easy to use application but also a powerful application.

2. LISTA DE PALABRAS CLAVE

Videojuego

Aprendizaje

Diseño

Herramienta educativa

2. LIST OF KEYWORDS

Videogame

Learning

Design

Educational tool

INDICES

Índice completo del contenido con números de página

Índice completo de figuras con números de página.

3. GLOSARIO

API – Proviene de las siglas de "Application Programming Interface", se trata de un conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software.

Bucle de juego – Se trata de una serie de métodos que se pueden sobrecargar y que el juego implementa para facilitar la comunicación entre el juego y el sistema operativo.

Framework - Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Motor de videojuegos – Se trata de una aplicación o una serie de rutinas que permiten el diseño, la creación y la representación de un videojuego, suelen incluir funcionalidades como un motor de renderizado para gráficos, métodos de detección de colecciones etc.

NPC – El termino proviene de las siglas de non-player carácter, se trata de un personaje que forma parte del programa y que no está controlado por el jugador.

Sprite - Los sprites son mapas de bits en 2D que se utilizan para representar elementos de un juego, como por ejemplo personajes.

4. CUERPO DEL TFG

4.1 Introducción

En los últimos años se ha demostrado que los videojuegos pueden tener importancia más allá de lo lúdico, los videojuegos pueden ser potentes herramientas para el aprendizaje.

La principal ventaja que aportan los videojuegos respecto a las técnicas tradicionales de enseñanza es el aumento de motivación que provoca en la mayoría de estudiantes, el proceso de aprendizaje resulta más sencillo cuando la persona tiene que buscar información por su cuenta para buscar soluciones a un determinado problema o a un juego que cuando tiene que simplemente memorizar fragmentos de un texto.

Una vez observada la gran importancia que pueden tener los videojuegos en el campo del aprendizaje, surge el gran problema al que se intenta dar solución con este TFG. Anteriormente para poder utilizar el diseño de videojuegos como herramienta de aprendizaje era necesario que la persona tuviera unas ciertas nociones de programación o al menos una comprensión del funcionamiento del juego a un bajo nivel, por lo que eran necesarios unos conocimientos informáticos relativamente avanzados.

El principal objetivo que persigue este trabajo es permitir la creación de videojuegos didácticos de una manera sencilla a personas que no disponen de conocimientos de programación, es decir subir el diseño del videojuego a un nivel más alto, por ejemplo el usuario introducirá un personaje en el videojuego de una manera visual y sencilla y la aplicación se encargara de generar el código correspondiente para que dicho personaje tenga el funcionamiento esperado.

Respecto a la estructura de la memoria, se ha optado por seguir el modelo recomendado añadiéndole algunos apartados nuevos para aclarar algunos aspectos del proyecto que se han considerado claves:

- Resumen del TFG – Breve resumen de la motivación y estructura del proyecto realizado. Este apartado de la memoria se incluye en castellano y en Ingles.

- Lista de Palabras clave – Palabras que serán utilizadas la para catalogación bibliográfica. Este apartado de la memoria se incluye en castellano y en Ingles.

- Índices – Se han incluido dos índices:

    - Índice completo del contenido con números de página.

    - Índice completo de figuras con números de página.

- Glosario

- Cuerpo del TFG – En el que se distinguen los siguientes apartados:

    - Introducción – Motivación y objetivos del TFG, también se incluye en este apartado la estructura de la memoria realizada.

    - Estudio del estado del arte o tecnologías a utilizar

    - Diseño y desarrollo

    - Principales retos encontrados – Principales dificultades que se han encontrado durante el desarrollo del proyecto y las soluciones para esas dificultades.

    - Pruebas y resultados

    - Sugerencias y soluciones – En este apartado se tratan las sugerencias que se recibieron del cliente cuando este probó la aplicación

(personas del departamento de Historia de la Universidad Autónoma de Madrid) y las mejoras implementadas en la aplicación para dar respuesta a estas.

- Conclusiones y trabajo futuro

- Referencias

- Anexos técnicos – Concretamente se han adjuntado dos anexos: el informe de las pruebas unitarias del sistema y el manual de usuario de la aplicación.

4.2 Estudio del estado del arte o tecnologías a utilizar

Antes de introducirnos en el desarrollo del proyecto es importante estudiar los diferentes aspectos y las diferentes tecnologías sobre las que se sustenta este proyecto, en primer lugar se tratara el tema de los videojuegos y de la educación y después se realizara una pequeña introducción a las tecnologías que se han utilizado en este proyecto, concretamente XNA y el lenguaje Visual Basic.

4.2.1 Videojuegos y educación

Un videojuego es un programa informático interactivo destinado al entretenimiento que puede funcionar en diversos dispositivos.
Los videojuegos están muy presentes en la sociedad actual, de hecho en los últimos años la industria de los videojuegos ha superado a la del cine y se ha convertido en la industria del ocio que más dinero mueve.

Aunque en un principio los videojuegos fueran diseñados como un elemento meramente lúdico, en los últimos años se han abierto posibilidades de aplicación de estos en el ámbito educativo.
Dentro del campo educativo, la enseñanza mediante el uso de videojuegos se ha demostrado bastante eficaz, llegándose incluso en algunos casos a superar a los métodos de enseñanza tradicionales.
Se puede observar en el estudio “Un Estudio Diferencial de las Calificaciones de Estudiantes Universitarios que han basado su aprendizaje en el Diseño de Videojuegos” [1] que los alumnos que optan por utilizar los videojuegos como herramienta educativa tienden a obtener unas mejoras marcas académicas, además dichos alumnos obtienen unos valores más altos de motivación.

En el estudio “Profesores frente a los videojuegos como recurso didáctico” [2] se puede observar que no solo los alumnos experimentan un cambio al usar los videojuegos como una herramienta educativa, sino que también los docentes cambian radicalmente su actitud respecto a los videojuegos como herramientas educativas al ver los resultados obtenidos.

¿Por qué los videojuegos son una herramienta de enseñanza tan potente? La principal ventaja con la que parten los videojuegos educativos respecto a las herramientas educativas tradicionales es la mayor motivación que generan en los alumnos.
Lograr que un alumno se sienta o no motivado a aprender "algo" es una de las claves del aprendizaje autónomo [3].

Desde el punto de vista de la teoría del aprendizaje social, algunos de los factores que fomentan la motivación durante el aprendizaje son:

- Carácter lúdico de los aprendizajes – Es decir, que el proceso de aprendizaje sea divertido para el alumno.

- Dificultad creciente y progresiva de las habilidades, pero adaptada al ritmo de cada uno, posibilidad de repetir y corregir los errores – Esto es especialmente importante para evitar que las personas que están realizando el proceso de aprendizaje no se aburran porque este resulte muy sencillo o muy difícil.

- Estimulación simultánea a múltiples niveles: visual, auditivo, etcétera.

- Identificación con héroes o personajes que fomentan la imitación.

Como se puede observar la mayoría de estos factores están presentes en los videojuegos y es esto lo que los hace una herramienta tan potente en el campo educativo si se usan correctamente.

Una vez observado el potencial de los videojuegos en la educación, es importante averiguar qué aspectos del individuo pueden ser desarrollados por los videojuegos educativos.
Según el estudio "Videojuegos y educación: Revisión crítica de la investigación realizada" [4] los videojuegos pueden ayudar a mejorar las siguientes capacidades de los individuos:

- Lectura – Un videojuego puede estimular la lectura de libros relacionados con el tema del mismo.
- Pensamiento lógico – Los videojuegos estimulan al usuario a desarrollar un pensamiento lógico para entre otras cosas dar solución a problemas.
- Observación.
- Vocabulario – Los videojuegos permiten a los usuarios aprender términos que anteriormente desconocían.
- Conocimiento básico – Permiten al usuario adquirir conocimientos o habilidades necesarias para su vida cotidiana.
- Ortografía
- Planificación de estrategias.

Con todo lo mencionado anteriormente, se puede observar el gran potencial que tienen los videojuegos en el campo de la educación.

4.2.2 XNA

Una de las tecnologías sobre las que se ha basado este proyecto ha sido XNA, concretamente todo lo referente al motor configurable de videojuegos se ha desarrollado sobre XNA 4.0.

XNA es un conjunto de herramientas desarrolladas por Microsoft que proporcionan una API para el Desarrollo de Videojuegos para diversas plataformas como pueden ser Windows o Xbox 360. La última versión (que es la que se ha utilizado en el proyecto) fue publicada por Microsoft en marzo de 2010 e incluyo entre otras cosas soporte para Windows Phone.

Los videojuegos realizados en XNA tienen un bucle de juego que sigue el siguiente esquema:

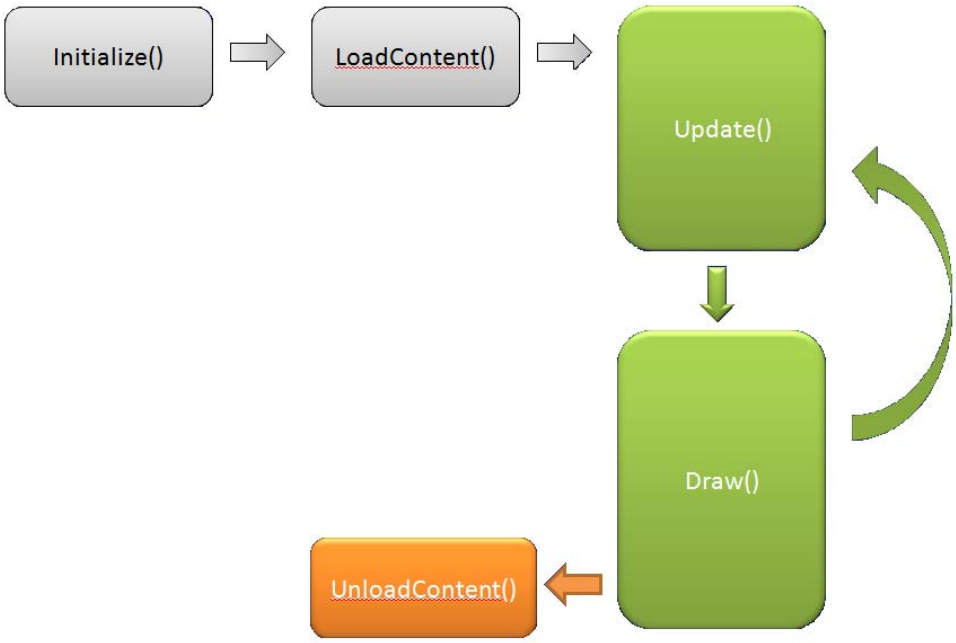


Figura 1. Bucle de juego de XNA

En primer lugar se inicializan los valores requeridos (Initialize), después se cargan los diferentes componentes del juego (LoadContent), como por ejemplo las texturas y se entra en un bucle en el que se va interactuando con el usuario, actualizando el mundo del juego y dibujándolo en pantalla (Update y Draw), por ultimo cuando el juego se termina se liberan los recursos reservados (UnloadContent)

Pese a no ser el motor de videojuegos más actual del mercado, se ha optado por utilizar XNA porque este presenta diversas características que hacen que sea idóneo para este tipo de proyectos, como por ejemplo:

- En primer lugar XNA se ejecuta en un entorno denominado XNA Framework que hace de intermediario entre el diseñador del videojuego y el propio sistema, permitiéndonos utilizar los diferentes recursos del sistema sin tener que conocer todos sus entresijos y sin tener que gestionar dichos recursos.

- XNA es totalmente gratuito, fácil de descargar y fácil de utilizar.

- Por ultimo XNA tiene unos requisitos mínimos bastante reducidos por lo que es posible utilizarlo en prácticamente cualquier máquina.

- La comunidad de desarrolladores de XNA es muy grande, por lo que es relativamente sencillo encontrar soluciones a problemas que pudieran surgir durante el desarrollo del programa.

4.2.2 Visual Basic

Por otra parte, la aplicación encargada de interactuar con el usuario y de hacer de intermediario entre dicho usuario y el motor de videojuegos se ha desarrollado con el lenguaje de programación Visual Basic 6.0.

Visual Basic es un lenguaje de programación desarrollado por la empresa Microsoft cuya primera versión fue presentada en 2001, en este proyecto utilizaremos la versión más reciente, que es la 6.0.

Las principales razones que me han llevado a utilizar este lenguaje en el desarrollo de la aplicación han sido:

- Visual Basic integra la implementación de los formularios de Windows y tiene acceso a gran parte de la API de Windows.

- Se trata de uno de los lenguajes de uso más extendido por lo que es fácil encontrar información y librerías que pueden resultar útiles en el desarrollo del proyecto.

4.3 Diseño y desarrollo

Dentro de este proyecto se puede distinguir dos aplicaciones que interactúan entre sí pero que son muy diferentes:

- Motor de videojuegos – Se trata básicamente de una herramienta creada sobre el motor X.N.A 4.0 que puede ser configurada mediante unos determinados archivos de configuración para generar diferentes videojuegos.

- Aplicación encargada de crear juego – Se trata de una aplicación que permite al usuario modificar los archivos de configuración con los que se generara el juego de una manera sencilla, básicamente la aplicación ocupa una capa intermedia entre el usuario y el motor configurable. Esta aplicación se ha programado con Visual Basic 6

A continuación se pasara a explicar de una manera más detallada el diseño de las aplicaciones finales desarrolladas y los diferentes módulos de dichas aplicaciones.

4.3.1 Diseño detallado

Al tratarse de dos aplicaciones bastante diferentes entre sí, he optado por realizar esta parte de la memoria por separado para cada aplicación, en primer lugar se tratara el motor configurable de videojuegos diseñado y a continuación se tratara la aplicación encargada de crear los videojuegos.

Motor de videojuegos

El motor de videojuegos desarrollado permite la creación de videojuegos de temática variada (aunque el objetivo que se persigue es que sea utilizado para crear juegos educativos) en 2 dimensiones.

El motor puede ser configurado mediante unos archivos de configuración para crear sus propios niveles unidos entre sí, sus propios personajes, sus propios objetos e incluso sus propias reglas complejas de funcionamiento del mundo del juego.

Este motor no solo permite al usuario crear este tipo de elementos, sino que también incluye las físicas y las funcionalidades necesarias para poder crear juegos en los que se pueda interactuar con dichos elementos de una manera fluida e intuitiva.

Además de los elementos y funcionalidades, el motor también puede ser configurado en el aspecto visual por el usuario para dotarlo del aspecto que se desee.

Una vez explicado el diseño del motor de una manera general, se va a pasar a explicar las clases que forman parte de dicho motor.

A grandes rasgos la aplicación está formada por una "gran" clase videojuego que contiene colecciones de las otras clases (Espacio, Objeto, NPC, Uniones, Reglas y Conversaciones).
El diagrama de clases de esta aplicación juego sería el siguiente:

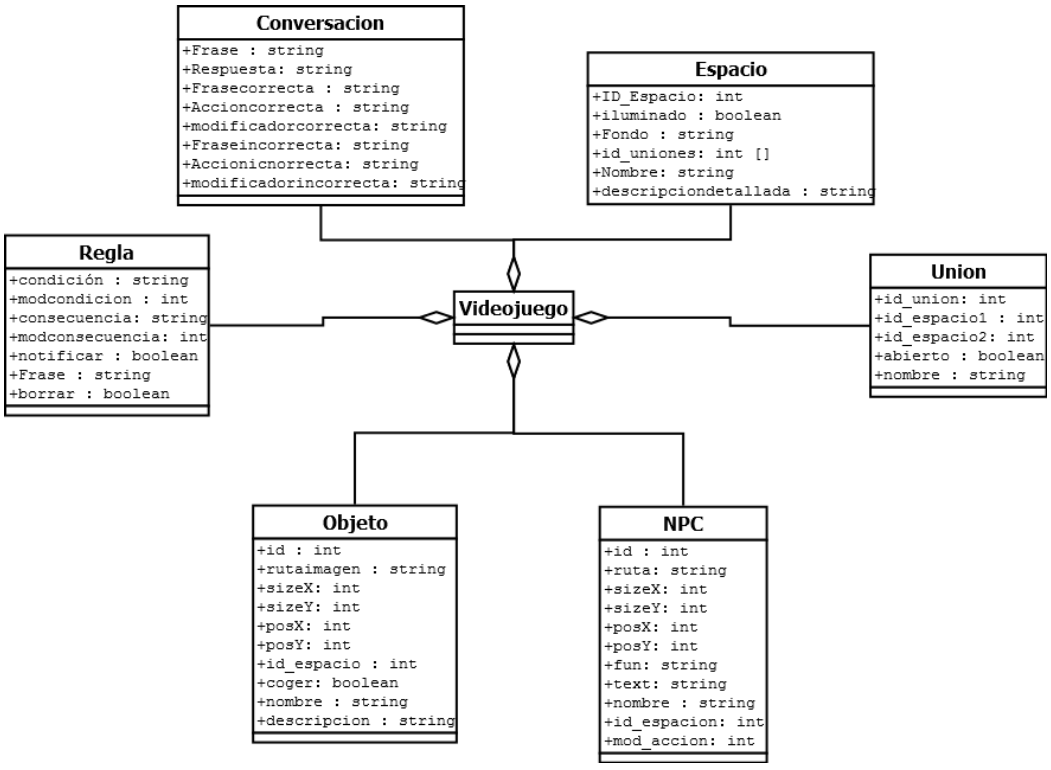


Figura 2. Diagrama de clases del videojuego

En el diagramas no se han incluido las funciones de las diferentes clases para simplificarlo, en los siguientes apartados se explicaran las funciones más importantes de cada clase.

A continuación pasare a explicar con más detalle cada una de las clases del motor configurable, tratando de aclarar el concepto detrás de la clase, los atributos de la misma y sus funciones más importantes.

Espacio

Un espacio básicamente es un nivel del juego por el que el usuario puede moverse libremente y en el que pueden estar situados diversos elementos, como por ejemplo objetos o NPCS (Personajes no controlados por el usuario).
Dentro de un espacio, el usuario puede realizar diferentes acciones como por ejemplo examinar el espacio para obtener una breve descripción de este, coger un objeto, hablar con un personaje etc.
Un espacio además puede o no estar iluminado con las diferentes consecuencias que eso pudiera provocar en el juego, como por ejemplo no poder ver nada al entrar al espacio, o no poder ver/coger objetos.

Respecto al diseño de la clase que contiene el TAD Espacio y todas las funcionalidades necesarias para tratar y usar dicho dato, hay que mencionar que el TAD contiene los siguientes atributos:

- int ID_espacio – Identificador numérico único para cada espacio.
- boolean iluminado – Indica si un espacio está iluminado o no.
- string Fondo – Ruta de la imagen de fondo del espacio
- int id_uniones[] – Identificadores de las uniones que comunican con el espacio, las uniones están explicadas en la pág. 19 de la memoria.
- string nombre – Nombre del espacio
- string descripciondetallada – Descripción que obtendrá el usuario cuando examine el espacio

Además de definir el TAD espacio, se han implementado las siguientes funciones:

Función: *Espacio*
Descripción: Constructor de la clase espacio
---
public Espacio(int id_espacio, Boolean iluminado, string fondo, int id_union_norte, int id_union_sur, int id_union_este, int id_union_oeste, string nombre,string descripciondetallada)

Función: *IniarrayEsp*
Descripción: Función encargada de inicializar una lista de espacios
---
public int IniarrayEsp(Espacio[] array, int tam)

Función: *DevuelveFondo*
Descripción: Devuelve el nombre de la imagen de fondo
---
public String DevuelveFondo(Espacio[] array, int id_espacio)

Función: *IluminaEspacio*
Descripción: Se encarga de iluminar un espacio
---
public int IluminaEspacio(Espacio[] array, int id_espacio)

Función: *ApagaEspacio*
Descripción: Se encarga de apagar un espacio
---
public int ApagaEspacio(Espacio[] array, int id_espacio)

Función: *Obtieneidnorte*
Descripción: Obtiene la ID de la unión situada al norte
---
public int Obtieneidnorte(Espacio[] array, int id_espacio)

Función: *Obtieneidsur*
Descripción: Obtiene la ID de la unión situada al sur
---
public int Obtieneidsur(Espacio[] array, int id_espacio)

Función: *Obtieneideste*
Descripción: Obtiene la ID de la unión situada al este
---
public int Obtieneideste(Espacio[] array, int id_espacio)

Función: *Obtieneidoeste*
Descripción: Obtiene la ID de la unión situada al oeste

---
public int Obtieneidoeste(Espacio[] array, int id_espacio)

Objeto

Un objeto es un elemento que puede ser manipulado por el usuario, principalmente el usuario puede realizar cuatro acciones con los objetos: coger el objeto, soltar el objeto, examinar el objeto y utilizarlo (si fuera posible).
Además cuando el usuario coge un objeto se pueden producir eventos de lo más diverso en el mundo, como por ejemplo abrirse una determinada puerta al coger un amuleto del suelo.

Un objeto puede estar contenido en un determinado espacio, en el propio inventario del jugador o bien no estar contenido en ningún lugar concreto por ejemplo esperando a que se cumpla una cierta condición para aparecer en algún lugar.

Respecto al diseño de la clase que contiene el TAD Objeto, hay que destacar que el TAD contiene los siguientes atributos:

- int id – Identificador numérico único para cada objeto
- string rutaimagen – Ruta de la imagen del objeto
- int sizeX- Ancho de la imagen del objeto(en pixeles)
- int sizeY – Alto de la imagen del objeto(en pixeles)
- int posX – Coordenada X del objeto (en pixeles)
- int posY – Coordenada Y del objeto(en pixeles)
- int id_espacio – Identificador del espacio en el que se encuentra el objeto.
- boolean coger – Indica si un objeto puede ser recogido por el jugador.
- string nombre – Nombre del objeto
- string descripcion – Informacion que obtendrá el usuario al examinar el objeto

Además de definir el TAD objeto, se han implementado las siguientes funciones:

Función: Objeto
Descripción: Constructor de la clase objeto
---
public Objeto(int id, String ruta, int sX, int sY, int pX, int pY, int id_esp, Boolean coger, String nombre, String descripcion)

Función: IniarrayObj
Descripción: Función encargada de inicializar una lista de objetos
---
public int IniarrayObj(Objeto [] array,int tam)

Función: Obtienedescripcion
Descripción: Devuelve la descripción de un objeto
---
public String Obtienedescripcion(Objeto[] array, String nom,int tam)

Función: *calculaid*
Descripción: Devuelve la ID de un objeto a partir de un nombre
---
public int calculaid(Objeto[] array, String nom, int tam)

NPC

Un NPC (non-player character) es un personaje controlado por el sistema con un comportamiento autónomo y con el que el jugador puede interactuar.

En el juego se pueden configurar los NPCs para que interactúen con el usuario de un gran número de formas diferentes, desde interacciones simples como por ejemplo dar un objeto al usuario hasta conversaciones complejas en las que se interactúa con la entrada de texto por teclado por parte del usuario.
Además el tipo de interacción del NPC con el jugador puede variar en tiempo de ejecución en función de diversas condiciones, por ejemplo un NPC puede interaccionar con el usuario de una manera diferente si este tiene un determinado objeto en el inventario o no.

En lo referente al diseño de la clase que contiene el TAD NPC, se han incluido los siguientes atributos para el TAD:

- int id – Identificador numérico único para cada NPC
- string ruta –Ruta de la imagen del NPC
- int sizeX – Ancho de la imagen del NPC (en pixeles)
- int sizeY– Alto de la imagen del NPC (en pixeles)
- int posX - Coordenada X del NPC (en pixeles)

- int posY - Coordenada Y del NPC (en pixeles)
- string fun – Identificador de la función que se ejecutara cuando el usuario interaccione con el NPC.
- string Text – Frase asociada a la función realiza por el NPC si la hubiera.
- string nombre – Nombre del NPC
- int id_espacio – Identificador del espacio en el que se encuentra el NPC.
- int mod_accion – Modificador de la acción que realiza el NPC si ese fuera necesario, es decir si la acción fuera "Dar objeto", mod_accion sería igual al identificador del objeto que se entregara al usuario.

Además de definir el TAD NPC, se han implementado las siguientes funciones:

Función: npc
Descripción: Constructor de la clase npc
---
public npc(int id,String ruta, int sX, int sY, int pX, int pY, String fun, String Text, String nombre, int id_espacio, int mod_accion)

Función: Iniarraynpc
Descripción: Función encargada de inicializar una lista de npc
---
public int Iniarraynpc(npc [] array,int tam)

Función: *Nuevaaccion*
Descripción: Se encarga de cambiar la acción de un personaje
---
public int Nuevaaccion(npc[] array, int tam, int id_npc, String accion, String modificador, String Frase)

Unión

Una unión es una conexión entre dos espacios diferentes dentro de un mismo juego.

Las uniones permiten al usuario moverse entre los diferentes espacios del sistema y pueden estar temporalmente cerradas (no permiten el paso del jugador a través de ellas) hasta que se produzca un determinado evento dentro del juego, como que por ejemplo el jugador utilice una llave para abrir la unión.

En lo referente al diseño de la clase que contiene el TAD Unión, es importante mencionar que el TAD contiene los siguientes atributos:

- int id_union – Identificador numérico único para cada unión
- int id_espacio1 – Identificador numérico del primero de los espacios que conecta esta unión.
- int id_espacio2 – Identificador numérico del segundo de los espacios que conecta esta unión.
- boolean abierto – Indica si una unión permite el paso al usuario o si por el contrario la unión esta bloqueada.
- string nombre – Nombre de la unión

Además de definir el TAD Unión, se han implementado las siguientes funciones dentro de la clase unión:

Función: Union
Descripción: Constructor de la clase Union
---
public Union(int id_union, int id_espacio1, int id_espacio2, Boolean abierto,String nombre)

Función: IniarrayUni
Descripción: Función encargada de inicializar una lista de uniones
---
public int IniarrayUni(Union [] array,int tam)

Función: Numespaciosiguiente
Descripción: Función que devuelve la ID del espacio que no es el actual del par de espacios que conforman la unión.
---
public int Numespaciosiguiente(Union[] array, int Id_union,int id_esp_actual)

Función: AbreUnion
Descripción: Función que se encarga de abrir una unión
---
public int AbreUnion(Union[] array, int Id_union)

Función: CierraUnion
Descripción: Función que se encarga de cerrar una unión
---
public int CierraUnion(Union[] array, int Id_union)

Regla

A grandes rasgos las reglas se pueden entender como unos cambios que se producen en el entorno del juego cuando se cumplen unas determinadas condiciones, por ejemplo una puerta que se abre cuando se usa una llave.

En esta clase se ha intentado realizar una implementación lo más genérica posible ya que las condiciones que pueden provocar que se ejecute una determinada regla y la acciones que realizan la reglas pueden ser de muy diverso tipo.

Dejando de lado las condiciones de ejecución de las reglas y las acciones que provocan en el juego cuando se ejecutan, las reglas pueden variar en otros dos aspectos: por un lado una regla puede avisar al usuario de su ejecución o no y por otro lado una regla puede ejecutarse cada vez que se cumpla la condición o bien una única vez y después borrarse.

En lo referente al diseño de la clase que contiene el TAD Regla, es importante mencionar que el TAD contiene los siguientes atributos:

- string condición – Acción que al cumplirse provoca que se ejecute la regla.
- int modcondicion – Modificador de la condición si esta fuera necesaria, por ejemplo si la acción que provoca que se ejecute la regla es “Entrar espacio”, modcondicion sería el identificador de ese espacio.
- string consecuencia – Acción que se ejecuta al cumplirse la condición de la regla.
- int *modconsecuencia* – Modificador de la consecuencia si esta fuera necesaria, por ejemplo si la acción que se produce al cumplirse la ejecución es “Abrir Union”, modconsecuencia sería el identificador de esa unión.
- boolean notificar – Indica si la ejecución de la regla se notifica al usuario o si por el contrario se ejecuta sin avisar.
- string Frase – Texto que se le muestra al usuario para avisarle de la ejecución de la regla.

- boolean borrar – Indica si la regla se ejecuta una vez y se elimina o si por el contrario la regla se ejecuta cada vez que la condición se cumple.

Además de definir el TAD Regla, se han implementado las siguientes funciones dentro de la clase regla:

Función: Regla
Descripción: Constructor de la clase Regla
---
public Regla(String condicion, int modcondicion, String consecuencia, int modconsecuencia, Boolean notificar,String Frase,Boolean borrar)

Función: IniarrayReg
Descripción: Función encargada de inicializar una lista de Reglas
---
public int IniarrayReg(Regla [] array,int tam)

Conversación

Una conversación es una sucesión de interacciones entre un NPC y el usuario, que fluye de una manera u otra en función del texto que introduce el usuario por teclado.

Una conversación estar formada por un serie de pasos. Un paso básicamente contiene una respuesta correcta (la respuesta que se espera recibir del usuario) y dos acciones a realizar (una si la respuesta recibida es la esperada y otra si no), estas acciones pueden ser de muy diverso tipo, por ejemplo ir a otro paso de la conversación, recibir un objeto etc.

Dentro de la clase conversación, están definidos dos TAD, por un lado Paso y por otro lado Conversación.

El TAD Paso contiene los siguientes atributos:

- string Frase – Frase que el NPC dice al usuario si este se le acaba de acercar por primera vez o si no ha introducido nada por teclado.
- string Respuesta – Frase que se espera recibir por parte del usuario como contestación.

- string Frasecorrecta – Frase que se dirá al usuario si este ha introducido la respuesta esperada.
- string Accioncorrecta – Acción que se realizara si el usuario ha introducido la respuesta esperada.
- string Modificadorcorrecta – Modificador de la acción a realizar si el usuario ha introducido la respuesta esperada.
- string Fraseincorrecta – Frase que se dirá al usuario si este ha introducido una respuesta diferente a la esperada.
- string Accionincorrecta – Acción que se realizara si el usuario ha introducido una respuesta diferente a la esperada.
- string Modificadorincorrecta – Modificador de la acción a realizar si el usuario ha introducido una respuesta diferente a la esperada. necesario.

El TAD Conversación contiene los siguientes atributos:

- string nombre – Nombre que identificara a la conversación dentro del sistema.
- int num_pasos – Numero de pasos de los que consta la conversación
- int paso_actual – Numero de paso en el que se encuentra la conversación actualmente.
- Paso[] pasos – Colección de los pasos que forman parte de la conversación.

Las funciones implementadas en la clase Conversaciones han sido las siguientes:

Función: Conversacion
Descripción: Constructor de la clase Conversacion
---
public Conversacion(String nombre,int num_pasos,int paso_actual)

Función: IniarrayConversacion
Descripción: Función encargada de inicializar una lista de Conversaciones
---
public int IniarrayConversacion(Conversacion [] array,int tam)

Función: aniadePaso
Descripción: Función encargada de añadir un paso a una conversacion
---

public int aniadePaso(Conversacion[] array, Paso paso, String nombre,int tam,int num_paso)

Función: buscaConversacion
Descripción: Función encargada de buscar una conversación por nombre
---
public Conversacion buscaConversacion(Conversacion[] array,string nombre, int tam)

Función: cambiapasoConversacion
Descripción: Función encargada de cambiar el paso actual de la conversacion
---
public int cambiapasoConversacion(Conversacion[] array, string nombre, int tam,int nuevopaso)

Videojuego

La clase Videojuego utiliza todas las clases que se han ido desglosando en los apartados anteriores para generar y gestionar el videojuego que se haya diseñado.

Dentro del gran número de funciones que se han implementado dentro de esta clase se distinguen dos tipos básicos de funciones:

- Funciones de carga – Se trata de las funciones encargadas de leer los archivos de configuración del motor y de crear un videojuego a partir de los valores leídos. Básicamente estas funciones cargan el contenido de los archivos de configuración a las estructuras correspondientes de la memoria y generan el juego deseado.

- Funciones de gestión – Se trata de las funciones que se utilizaran durante la ejecución del juego como por ejemplo la función encargada de gestionar las colisiones entre objetos o la función encargada de comprobar si se ha cumplido la condición de una regla.

Algunas de las funciones más importantes de la clase Videojuego son:

Función: LeeDatosJuego
Descripción: Función encargada de leer todos los datos de los archivos de configuración y cargarlos en memoria para poder crear el juego
---
Public int LeeDatosJuego(String rutaDirectorio)

Función: GestionaReglasObjetosJuego
Descripción: Función encargada de gestionar todas las reglas del juego referentes a objetos, es decir las reglas que tienen como condición de ejecución acciones relacionadas con objetos, como por ejemplo coger objeto, examinar objeto etc.

Public int GestionaReglasObjetosJuego(Regla[] r,int num_reglas,int id_obj)

Función: GestionaReglasEspaciosJuego
Descripción: Función encargada de gestionar todas las reglas del juego referentes a espacios, es decir las reglas que tienen como condición de ejecución acciones relacionadas con espacios, como por ejemplo entrar en un espacio, iluminar un espacio etc.

Public int GestionaReglasObjetosJuego(Regla[] r,int num_reglas,int id_obj)

Función: CalculaCogerObjetos
Descripción: Función encargada de gestionar la acción del usuario de coger un objeto, para ello comprueba que el objeto este cerca del usuario y que dicho objeto pueda ser recogido
---
public int CalculaCogerObjetos(Vector2 pos, String direccion, Objeto[] arrayobj)

Función: Calculadirecciones
Descripción: Esta función se encarga de indicar en la pantalla a que direcciones se puede mover el usuario desde el espacio actual , para ello estudia todas las uniones que conectan al espacio actual.
---
public void Calculadirecciones()

Función: ColisionesPixel
Descripción: Esta función devuelve si existe una colisión entre dos elementos a nivel de pixel, para ello hay que pasarle los tamaños de ambos elementos y los conjuntos de bits que forman cada una de las imágenes que representan los elementos, este conjunto de bits es necesario para gestionar las colisiones a nivel de pixel.
---
public static bool ColisionesPixel(Rectangle elemento1, Rectangle elemento2, Color[] bitsA, Color[] bitsB)

Función: CalculaColisionRectangulo
Descripción: Esta función devuelve si existe una colisión entre el jugador y alguno de los elementos del juego (ya sean NPCS u objetos). En primer lugar se observa si se produce una colisión en los rectángulos (ancho x alto de las imágenes) y si se produce una colisión a nivel de rectángulos, se comprueba la colisión a nivel de bits para evitar colisiones en zonas transparentes de las imagenes.
---
public int CalculaColisionRectangulo(Vector2 pos, String direccion, Objeto[] arrayobj)

Función: Videojuego
Descripción: Básicamente se trata del constructor de la clase, se encargar de realizar las llamadas a las funciones de carga y de inicializar el juego.

---
public Videojuego()

Función: LoadContent
Descripción: Esta función se encargar de cargar todas las texturas necesarias para el funcionamiento del juego, en el caso de que se produzca algún error en la carga de texturas aborta la ejecución y avisa al usuario del problema.
---
protected override void LoadContent()

Función: Update
Descripción: Esta función se ejecuta periódicamente mientras el juego se esté ejecutando, básicamente es la función que interactúa con el usuario (ya sea por teclado o por ratón) y que se encarga de realizar las llamadas a las funciones necesarias en función de las interacciones del usuario.
---
protected override void Update(GameTime gameTime)

Aplicación creadora de videojuegos

La principal función de la aplicación es permitir al usuario configurar los diferentes aspectos de su videojuego y guardar los datos de este videojuego en los archivos de configuración del sistema, de manera que luego puedan ser leídos por el motor para generar el videojuego deseado por el usuario.

Dentro de esta aplicación se pueden distinguir dos módulos principales con unas funcionalidades bastante diferenciadas:

- Por una parte se encuentra el modulo que se encarga de transformar las imágenes del usuario (formato .bmp, .jpg o .png) al formato reconocido por el motor de juego (.xnb). Este módulo permite al usuario introducir al sistema sus propias imágenes para que sirvan de modelo para personajes, objetos o niveles.

- Por otro lado se encuentra el modulo encargado de realizar el videojuego en sí, este módulo pide los datos al usuario y realiza las modificaciones oportunas en los archivos de configuración del motor para que este genere el juego deseado.

Una vez terminado el desarrollo del videojuego, la aplicación permite guardar todos los datos referentes al juego en un formato instalable, de manera que un usuario que hiciera un juego podría compartirlo con otras personas de una manera fácil y rápida.

Dentro del módulo de la aplicación encargada de modificar los archivos de configuración, existen diversos formularios encargados de realizar ciertas funciones en el sistema:

FormPrincipal

Se trata del formulario principal de la aplicación.
Desde este formulario se puede acceder a los dos grandes módulos mencionados anteriormente.

Además de permitir el acceso a los módulos, este formulario permite instalar las librerías que requiere la aplicación para funcionar correctamente, en concreto dichas librerías son:

- xnags_platform_tools.msi
- xnags_shared.msi
- xnafx40_redist.msi

Por último, este formulario también permite ejecutar una prueba del juego para ir probando todas los cambios introducidos por el usuario en el sistema hasta el momento, de manera que el diseñador del videojuego no necesita ir publicando su videojuego para probarlo, si no que puede ir ejecutando este juego de prueba y solo realizar la publicación del videojuego cuando el desarrollo de este haya terminado.

FormTransformadorImagenes

Se trata del módulo encargado de insertar las imágenes del usuario al sistema.

Para ello en primer lugar transforma las imágenes de entrada (.jpg, png y .bmp) al formato utilizable por el motor de videojuegos (.xnb) y en segundo lugar realiza los cambios necesarios en los ficheros de configuración para dejar constancia de la existencia de esa nueva imagen en el sistema y la información referente a esa imagen (tamaño, tipo etc.).

Desde este formulario se puede introducir cinco tipos de elementos al sistema:

- Fondos – Imágenes de fondo para espacios.

- Objetos – Imágenes que servirán de modelo para objetos del sistema.

- NPCS -Imágenes que servirán de modelo para personajes controlados por el sistema.

- Menús – Aspecto personalizado de los menús. Pese a incluir diversos formatos de Menús se ha optado por permitir a los usuarios introducir sus propias diseños de menús si estos lo desearan.
  Para facilitar este proceso de inserción de nuevas imágenes se adjunta junto a la aplicación unas plantillas con la forma de los diferentes menús del juego para que el usuario las modifique en vez de tener que diseñar las imágenes partiendo de cero.

- Pantallas de información – Se trata de la imagen que se mostrara al usuario cuando este pulse la opción “Saber más” de un elemento.

FormIntroducir

El formulario FormIntroducir es el formulario central del módulo encargado de pedir datos al usuario e introducirlos en los archivos de configuración.

Dentro de este formulario se pueden editar la mayoría de los datos del juego, está dividido en diferentes pestañas y desde cada una de las pestañas se pueden editar

diferentes aspectos del juego, concretamente se puede distinguir las siguientes pestañas dentro de este formulario:

- Personaje – Dentro de esta pestaña se puede elegir el aspecto visual del personaje que protagonizara el juego, además se puede observar una visualización del modelo del personaje.

- Espacios – Desde esta pestaña se puede configurar todo lo referente a los diferentes niveles del juego, desde el aspecto visual de los niveles hasta diferentes atributos de estos como pueden ser la iluminación o la descripción de estos.
  Además desde esta pestaña se nos permite acceder al formulario que muestra una visualización del mapa del juego (FormMaps) que estamos creando.

- Uniones – En este apartado del formulario se puede gestionar toda la información referente a las diferentes uniones que forman parte del juego.

- NPCs –En esta pestaña se permite crear nuevos personajes o borrar personajes del juego, prácticamente todos los aspectos de los personajes son configurables: la apariencia del personaje, el comportamiento de este con el usuario, el lugar donde se encuentra etc.

- Objetos – Dentro de esta pestaña se pueden crear o eliminar objetos del juego. Al igual que los NPCS, casi todos los aspectos de los objetos pueden ser configurados por el usuario, por ejemplo: la imagen del objeto, si puede ser movido, donde se encuentra etc.

- Conversaciones – En la pestaña de conversaciones se nos facilita el acceso a dos formularios diferentes: por un lado el formulario encargado de crear nuevas conversaciones y por otro lado el formulario encargado de borrar conversaciones creadas previamente. Se ha decidido gestionar la creación de conversaciones en un formulario aparte por ser este un proceso relativamente complejo.

- Reglas – En esta pestaña se pueden configurar los diferentes aspectos de las reglas del juego, como ya se comentó previamente en la memoria una regla es básicamente un acción que se ejecuta cuando se cumple una determinada condición durante la ejecución del juego.

- Aspecto visual – En este apartado se permite configurar los diferentes elementos visuales de nuestro juego, entre otras cosas se permite personalizar el aspecto de los diferentes menús del juego.

- Otros – En este apartado se incluyen las diferentes funciones que no tenían cabida en el resto de apartados, entre las funciones presentes en este apartado hay que destacar

    - Ver resumen – Permite ver todos los progresos realizados hasta el momento en el desarrollo del juego en un formato simple (texto).

    - Generar juego instalable – Permite transformar el juego que hemos ido creando a un formato instalable de manera que se puede enviar a otro usuario de una manera sencilla y cómoda.

    - Instalar juego – Permite instalar un juego en formato instalable en nuestra máquina, una vez instalado el juego, podemos o bien jugar a dicho juego o realizar modificaciones en el mismo.

    - Eliminar juego – Elimina todos los datos del juego, esto puede ser bastante útil por ejemplo si hemos terminado de diseñar un juego y queremos empezar a diseñar otro, ya que en vez de tener que ir borrando cada elemento uno a uno, podemos utilizar esta opción para eliminar todos.

FormConversaciones

Desde este formulario se nos permite editar los diferentes aspectos de las conversaciones de nuestro juego.
Como ya se mencionó anteriormente una conversación es una interacción compleja (más de un paso) entre el usuario y un personaje controlado por el sistema.
Una conversación está formada por un determinado número de pasos y cada uno de estos pasos contiene: una frase a decir, la respuesta que se espera recibir del usuario y dos acciones a realizar (una si la respuesta introducida por el usuario ha sido la esperada y otra si la respuesta recibida ha sido diferente a la esperada).

El principal motivo que me ha llevado a separar las conversaciones del resto de interacciones simples usuario-NPC ha sido que creo que para muchos juegos, estas interacciones complejas no se llegarían a utilizar por lo que en esos casos tener todo junto podría llevar a unas complicaciones innecesarias.
Al tenerlo separado del resto de interacciones, los usuarios que quieran realizar interacciones complejas pueden hacerlo, pero los usuarios que solo quieren hacer interacciones sencillas no necesitan acceder a este tipo de interacciones.

FormAyuda

Básicamente desde este formulario de la aplicación ser nos permite navegar por todos los contenidos de ayuda.

En función del formulario desde el que se haya llamado a la ayuda, se nos redirigirá a ciertos puntos del contenido de la ayuda, pero siempre se nos permitirá navegar libremente por esta.

FormMapa

En este formulario podemos ver una recreación del mapa del juego que vamos creando, se nos permite navegar por los diferentes espacios conectados entre sí y visualizar la información de dichos espacios.

FormCoordenadas

Este formulario se creó con la intención de facilitar al usuario la indicación de la posición que ocupan los NPCS y los objetos dentro de un espacio.

A la hora de insertar la posición de uno de estos elementos en un espacio, se da la opción al usuario de o bien insertar las posiciones (en pixeles) por teclado o utilizar este formulario para hacerlo de una manera más sencilla.

Para ello, este formulario muestra una recreación del espacio, con todos los objetos y personajes incluidos en el y permite seleccionar la posición donde se quiere realizar la inserción del elemento haciendo un simple clic.

4.3.2 Casos de uso

En este apartado de la memoria vamos a definir los principales roles de las personas que utilizaran la aplicación y las funcionalidades de la aplicación que utilizaran.
Se distinguen principalmente dos roles:

- Desarrollador – Se trata de la persona que diseña su propio videojuego, para ello accede a la aplicación y modifica los diferentes aspecto de su juego, una vez terminado puede crear un instalador para dar ese juego a otra persona.

- Jugador – Básicamente el rol del jugador consiste en realizar únicamente dos actividades, por una parte instalar juegos y por otra parte probar dichos juegos.

Los roles de los usuarios no son cerrados, en cualquier momento un jugador podría decidir desarrollar un juego y de esta manera convertirse en un desarrollador y de la misma manera un desarrollador puede acceder al sistema con un rol de jugador en cualquier momento y simplemente instalar y probar juegos desarrollados por otra persona.

El diagrama de los casos de uso de la aplicación seria el siguiente:

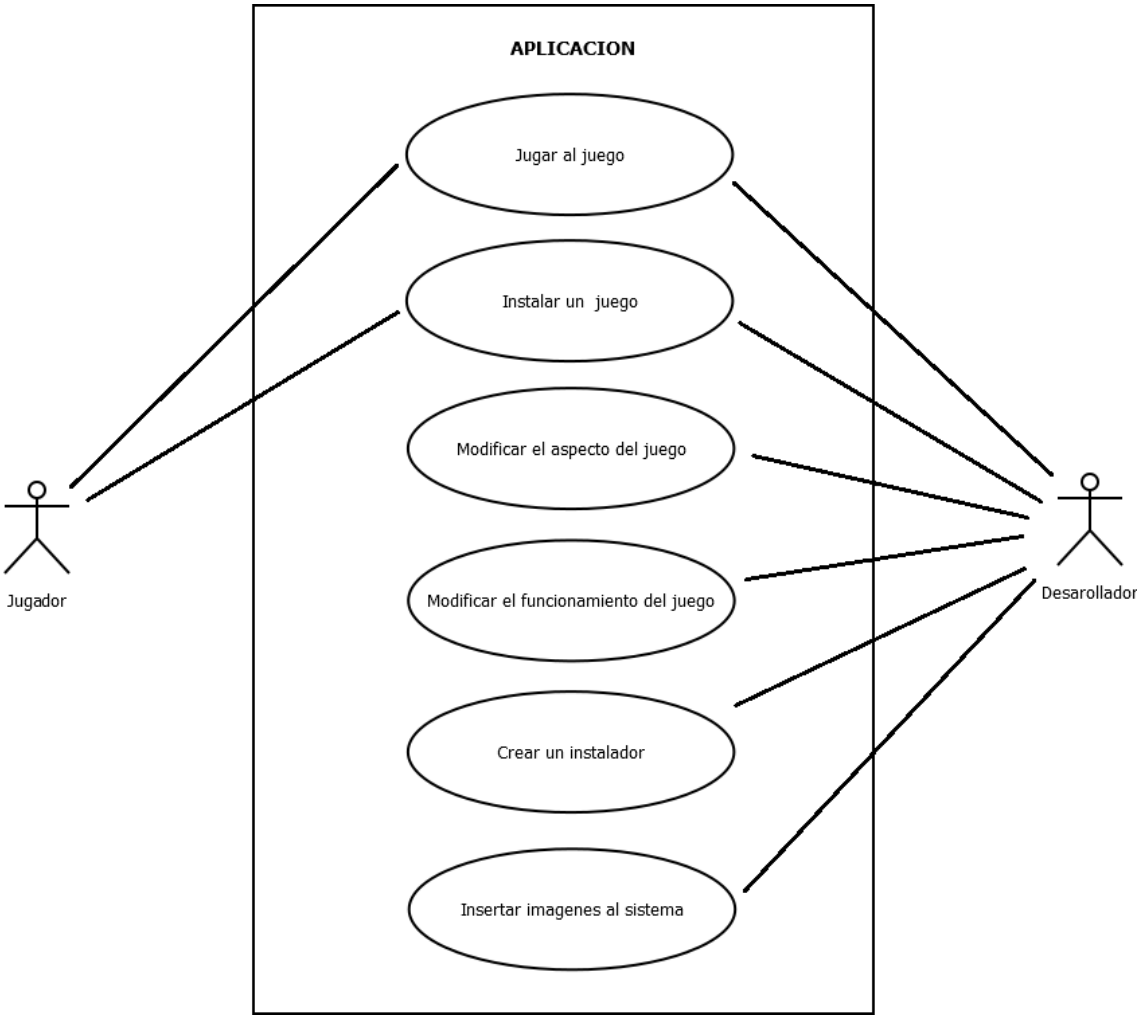


Figura 3. Diagrama de casos de uso

A continuación se va a proceder a explicar de una manera más detallada los distintos casos de uso mencionados anteriormente:

- Jugar al juego – Como su propio nombre indica, se trata simplemente de jugar a un juego creado anteriormente sin entrar a modificar ningún aspecto del mismo.

- Instalar un juego – Consiste en instalar un juego creado por otro usuario en nuestro sistema.

- Modificar el aspecto del juego – Consiste en realizar una serie de modificaciones al aspecto visual del juego, como puede ser cambiar el aspecto de los menús o el formato de las fuentes.

- Modificar funcionamiento del juego – Aquí se entraría en lo que viene siendo el diseño de juego en sí, es decir crear nuevos personajes, objetos, niveles etc.

- Crear un instalador – El desarrollador tiene la opción, una vez terminado el desarrollo de su juego, de crear un instalador para facilitar el intercambio del juego con otro persona.

- Insertar imágenes al sistema – El sistema permite al desarrollador insertar sus propias imágenes al sistema para que sirvan como modelo para objetos, personajes etc.

4.4 Principales retos encontrados

En este apartado de la memoria se va a tratar los principales problemas que se han encontrado durante el desarrollo de este proyecto y las soluciones que se han implementado para intentar resolver estos problemas de la mejor manera posible.

Crear una aplicación completa pero sencilla de usar

Uno de los principales retos que se ha presentado durante el desarrollo de este trabajo, ha sido el intentar conseguir un equilibrio entre la sencillez de uso de la aplicación y el número de opciones que se otorgaba al usuario para crear sus videojuegos.

Para intentar lograr este equilibrio se ha intentado que las opciones básicas de creación de un juego sean lo más sencillas posible y que sea el propio usuario de la aplicación el que decida acceder a las opciones más complejas del sistema si es lo que desea.

Otra de las acciones que se ha tomado para facilitar el uso de la aplicación, ha sido la creación de un sistema de ayuda que contiene instrucciones e información sobre todas las características de la aplicación.

Sistema de gestión de colisiones genérico, eficiente y preciso

Otro de los grandes retos que aparecieron durante la realización del trabajo fue la implementación de una función encargada de gestionar colisiones de una manera genérica para cualquier elemento que el usuario quisiera introducir en el sistema.

En un primer momento se optó por gestionar las colisiones simplemente con rectángulos del tamaño de las imágenes (ancho x alto) pero rápidamente se descartó porque podrían darse casos donde el usuario introduzca una imagen con mucha zona transparente lo que podría dar lugar a colisiones muy irreales desde el punto de vista del jugador.

La segunda solución a la que se llego fue el gestionar las colisiones entre elementos comparando pixel a pixel, esta solución pese a resolver el problema de las colisiones irreales, era extremadamente ineficiente ya que había que realizar una grandísima cantidad de operaciones para determinar si se estaba produciendo una colisión entre los diferentes elementos del sistema.

La solución final que se implementó para resolver este problema fue una combinación de las dos soluciones previas. Por una parte se comprueba la colisión entre los rectángulos del tamaño de las imágenes y en el caso de que efectivamente exista la colisión de rectángulos, se pasa a comprobar la colisión pixel a pixel entre los dos elementos.
De esta manera se consigue una gestión de colisiones realista pero ahorrándose una gran cantidad de operaciones innecesarias.

4.5 Pruebas y resultados

La estrategia que se ha seguido para realizar las pruebas que comprueben el correcto funcionamiento de la aplicación, se ha basado en la realización de tres niveles de pruebas, concretamente el proceso que se ha seguido para realizar las pruebas ha sido el siguiente:

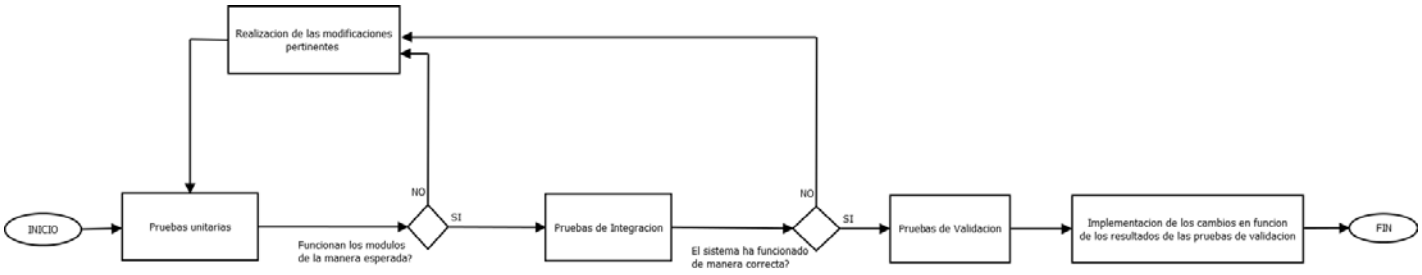


Figura 4. Diagrama del proceso de pruebas del proyecto

Cada uno de los niveles de pruebas ha tenido una función diferente:

- Pruebas unitarias – En primer lugar se han realizado baterías de pruebas unitarias para comprobar el correcto funcionamiento de cada uno de los elementos del sistema por separado.
  Los resultados de estas pruebas se pueden encontrar en el anexo 2 de este documento.

- Pruebas de integración – Una vez realizada la batería de pruebas unitarias, se pasó a realizar una serie de pruebas de integración del producto. Básicamente este tipo de pruebas consiste en probar todo el sistema software en su conjunto.
  Para ello se optó por crear varios videojuegos con mecánicas diferentes, dentro de estos videojuegos de pruebas realizados se puede destacar por ejemplo la creación de una nueva versión de “La Batalla de Kadesh”, un videojuego ambientado en el Antiguo Egipto que ya fue desarrollado anteriormente por un grupo de estudiantes de Historia con la ayuda de estudiantes de Informática.

- Pruebas de validación – Las pruebas de validación forman parte de un proceso en el que se comprueba que el software cumple con los requisitos requeridos. Dentro de las pruebas de validación que se han realizado se pueden distinguir dos subtipos:

    - Pruebas alfa – Pruebas realizadas por el propio desarrollador de la aplicación dentro de un entorno controlado.

    - Pruebas de aceptación – Concretamente se ha enseñado la aplicación a personas del departamento de Historia de la Universidad Autónoma de Madrid.

      En primer lugar se mostró al cliente (en este caso una persona del departamento de Historia) las posibilidades que ofrece la aplicación desarrollada y se realizó una prueba del software desarrollado.

      Una vez que el cliente hubo validado que la aplicación cumple con los requisitos esperados, se le pregunto al cliente qué cambios introduciría en la aplicación, de cara a estudiar posibles mejoras a implementar en la versión final.

      Tanto las sugerencias realizadas por el cliente, como las mejoras realizadas en la aplicación para dar solución a dichas sugerencias se pueden encontrar en el siguiente apartado de la memoria (Sugerencias y soluciones).

Lo que se ha perseguido con este plan de pruebas ha sido no solo comprobar y corregir posibles errores de la aplicación, sino que también se ha buscado realizar modificaciones en la aplicación basadas en las opiniones de personas externas al desarrollo del proyecto, buscando obtener de esta manera la mejor aplicación posible.

4.6 Sugerencias y soluciones

En este apartado de la memoria se van a tratar las sugerencias que realizo el cliente de cara a facilitar el uso de la aplicación o hacer la aplicación más completa, además se incluirá las soluciones que se dio a los problemas planteados y una breve explicación de dichas soluciones.
Las principales sugerencias que se recibieron fueron:

1ª Sugerencia

La sugerencia que se recibió fue sustituir los términos ambiguos o técnicos de la aplicación por otros más fáciles de entender por un usuario, por ejemplo cambiar el término NPC por Personajes etc.

Se optó por implementar esta sugerencia, tratando de cambiar todos los términos que pudieran llevar a confusión al usuario por otros más simples y sencillos de entender.

2ª Sugerencia

El cliente sugirió modificar los controles del juego para que sean más intuitivos. Concretamente se sugirió cambiar dos aspectos del control:

- En primer lugar cambiar la manera de controlar el personaje, concretamente manejar al personaje utilizando las flechas del teclado en vez de utilizar las teclas WASD.

- Permitir que el jugador pueda introducir texto por teclado pulsando la tecla “Enter” en vez de tener que hacer clic en un botón introducir.

Ambos cambios sugeridos fueron implementados en la versión final de la aplicación.

3ª Sugerencia

Se sugirió la posibilidad de añadir al juego la opción de transportar al jugador de un espacio a otro como consecuencia de algún evento, como por ejemplo usar un objeto. La idea que se perseguía era la de dotar de una mayor libertad de acción al diseñador de videojuego, para que este pudiera por ejemplo introducir un barco que transporte al jugador a una determinada isla, o un guía que te lleve a un determinado lugar etc.

Se decidido introducir esta posibilidad en la aplicación ya que se consideró que era una opción muy interesante que abría un amplio abanico de posibilidades a la hora de desarrollar un juego con la herramienta.

4ª Sugerencia

Se propuso implementar la posibilidad de quitar todos los objetos del inventario como consecuencia de algún evento, como por ejemplo entrar en un espacio.

En este caso se decidió no implementar esta sugerencia, porque el diseñador del videojuego ya tenía la posibilidad de implementar esta funcionalidad mediante el uso de varias reglas, y al tratarse de una acción relativamente “rara” y que podía implementarse con el uso de varias reglas, no merecía la pena añadir la opción.

5ª Sugerencia

Anteriormente, la aplicación solo permitía mostrar una pantalla en la que se mostraba un mensaje genérico en el que se felicitaba al usuario pon terminar el juego.
El cliente propuso añadir la posibilidad de que el diseñador del videojuego pueda personalizar dichas pantallas de juego, para que de esta manera se puedan implementar finales alternativos en un juego.
El cambio propuesto fue implementado en la versión final de la aplicación, de manera que ahora el diseñador del videojuego puede personalizar el tipo del final al que se ha llegado (el jugador ha ganado o perdido) y mostrar un mensaje personalizado al usuario.

6ª Sugerencia

Se propuso añadir la posibilidad de añadir pantallas de información adicionales para objetos o personajes.
La idea que se propuso era que el usuario al acercarse por ejemplo al personaje Ramsés II pudiera acceder a una pantalla que contuviera imágenes y textos sobre Ramsés.

Se decidió implementar esta sugerencia porque se consideró que con ella se podía incrementar bastante el nivel educativo de los videojuegos generados por la aplicación.

La implementación consistió en permitir al diseñador del videojuego introducir imágenes asociadas a objetos o personajes y que se mostraran cuando el jugador pulse "Saber más".

A continuación se incluye un ejemplo de pantalla de información, en este caso de trata de información acerca del faraón Ramsés II.

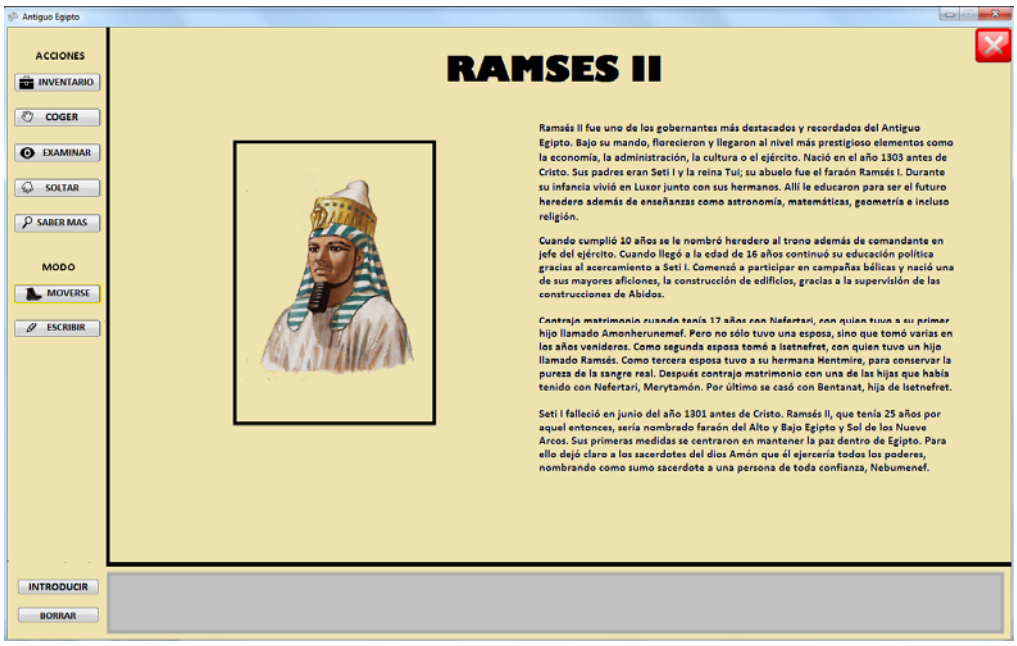


Figura 5. Pantalla de información adicional

7ª Sugerencia

Se sugirió añadir una mayor cantidad de información en las pantallas de ayuda y en los manuales de usuario, concretamente se propuso añadir información sobre los siguientes temas:

- Explicación de la vista que se sigue en el juego, para facilitar la creación de fondos, objetos y personajes al usuario.
- Tutorial en el que se crea un juego, en el que se usan la mayoría de opciones que permite la aplicación, pasó a paso.
- Información sobre algunos temas externos a la aplicación pero necesarios para el diseño de juegos, como por ejemplo como crear imágenes con transparencias de una manera sencilla.

Se añadió información sobre todos los temas sugeridos, ya sea en el manual de usuario o en el propio menú de ayuda de la aplicación.

8ª Sugerencia

Se propuso que la aplicación permitiera al usuario marcar grandes zonas de la pantalla como terreno impracticable, es decir terreno por el que el jugador no pueda caminar.
En un primer momento se intentó realizar esta función en la aplicación, pero se acabó descartando esta opción porque las formas de las zonas de una pantalla que el usuario no quiere que sean atravesadas puede ser muy diversas y casi nunca van a tener forma de polígonos regulares.
Finalmente se optó por añadir en el manual de usuario una explicación sobre cómo conseguir este efecto utilizando imágenes con transparencia.

4.7 Conclusiones y trabajo futuro

En este trabajo el principal objetivo que se perseguía era el de diseñar una aplicación que permitiera a usuarios con un escaso nivel de conocimientos informáticos diseñar sus propios videojuegos de temática educativa.
Creo que este objetivo se ha cumplido de una manera bastante satisfactoria, la aplicación tiene un uso bastante sencillo e intuitivo, por lo que la única dificultad que tendría una persona a la hora de crear un nuevo videojuego seria el propio diseño del videojuego y no cuestiones técnicas.

Otro de los objetivos que se perseguía en este trabajo, que también creo que se ha conseguido, era el de interactuar con personas de otros departamentos (en este caso con personas del departamento de Historia de la Universidad Autónoma de Madrid), durante el desarrollo de la aplicación de una manera efectiva, para que de esta manera la aplicación desarrollada sea verdaderamente útil y pueda ser usada en posteriores cursos por alumnos para realizar algún tipo de trabajo.

Respecto a posibles líneas futuras en las que sería interesante trabajar de cara a mejorar el trabajo creado, creo que sería interesante trabajar en los siguientes campos:

- En primer lugar creo que sería muy interesante utilizar la herramienta desarrollada con alumnos y medir el impacto que tiene el uso de esta aplicación en diferentes aspectos como la motivación o los resultados académicos de dichos alumnos.
  De esta manera se podría estudiar el impacto que el diseño de videojuegos puede tener en el campo del aprendizaje y de la motivación.

- Otro de los puntos en los que creo que sería interesante trabajar, sería en crear un repositorio online en el que se permita subir los juegos que diseñen los usuarios para que otras personas puedan descargarlos y disfrutar de ellos.

- Creo que también sería interesante estudiar la posibilidad de realizar una versión de la aplicación para dispositivos móviles.

- Por ultimo creo que sería interesante añadir nuevos comportamientos para los personajes controlados por el sistema, como por ejemplo personajes hostiles que intenten atacar al usuario.

En líneas generales, estoy bastante satisfecho con el trabajo realizado ya que se logró cumplir con los objetivos que se había propuesto. Además creo que he conseguido crear una herramienta que puede ser útil para futuros trabajos o investigaciones.

4.8 Referencias

[1] P. Molins-Ruano, C.Sevilla, S.Santini, P. Rodriguez y G. M. Sacha.
"Un Estudio Diferencial de las Calificaciones de Estudiantes Universitarios que han basado du aprendizaje en el Diseño de Videojuegos"

[2] Mercedes Leticia Sánchez Ambriz "Profesores frente a los videojuegos como recurso didáctico"

[3] Carina Soledad González González, Francisco Blanco Izquierdo "Emociones con videojuegos: Incrementado la motivación del aprendizaje"

[4] *Alfonso Méndiz,Julián Pindado,Javier Ruiz,José Mª* Pulido "Videojuegos y educación: Revisión crítica de la investigación realizada"

5. ANEXOS TECNICOS

Anexo I: Manual de usuario de la aplicación

INDICE

Instalación

Para la utilización del programa es necesario que estén instaladas en el sistema las siguientes librerías:

- xnags_platform_tools.msi
- xnags_shared.msi
- xnafx40_redist.msi

Para instalar dichas librerías en el sistema, hacemos clic en **INSTALAR XNA STUDIO**

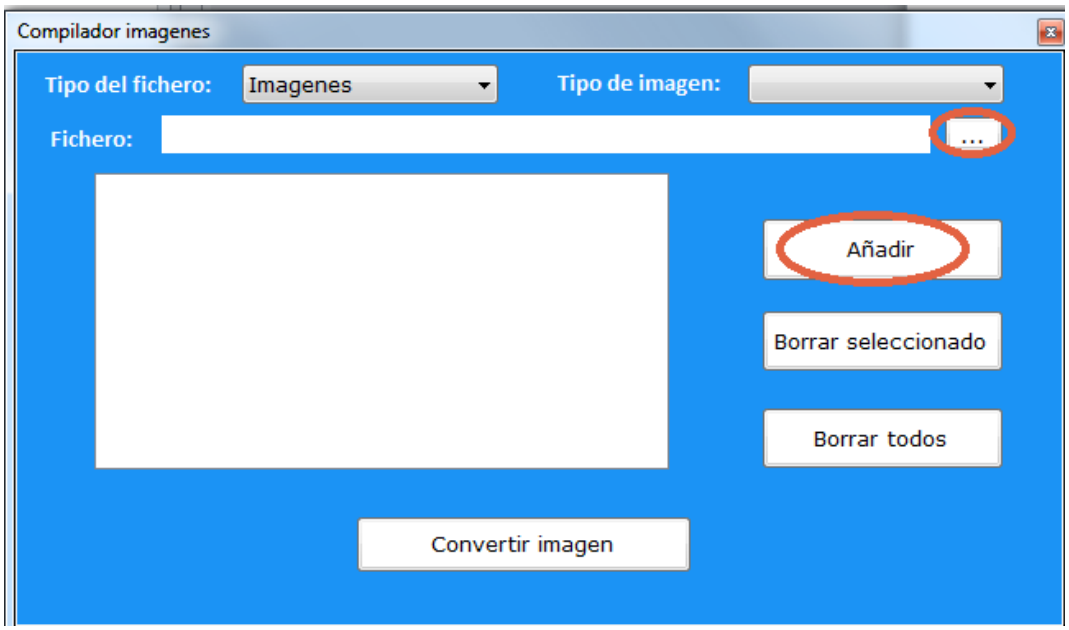
Al hacer clic en el botón, se nos abrirán los diferentes programas de instalación de los paquetes.
Una vez instalados los paquetes ya podemos ejecutar el programa con normalidad.

Introducir imagen en el Sistema

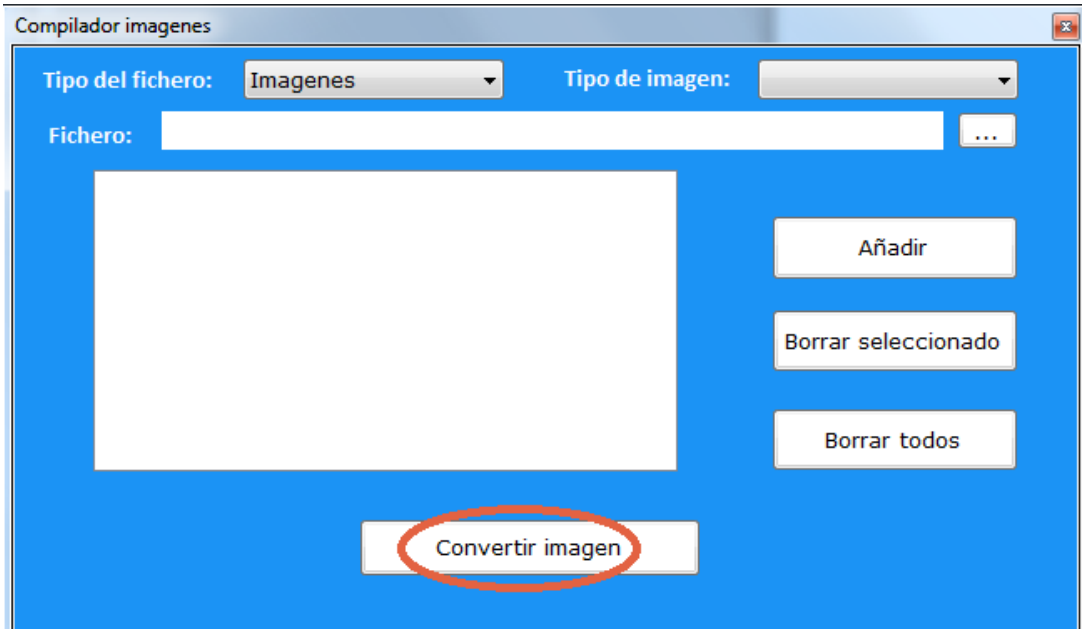
Para introducir una imagen como modelo (npc, objeto o fondo) en el Sistema, hacemos clic en el botón **TRANSFORMAR IMAGEN**

Al hacer clic en el botón se abrirá el programa encargado de la transformación de imágenes al formato requerido para su uso en el Sistema.
En primer lugar elegimos el tipo de dato que corresponde a la imagen, es decir si la imagen se trata de un objeto, un personaje etc.

Una vez elegido el tipo de objeto a insertar, elegimos la ruta donde se encuentra la imagen y lo añadimos a la lista de objetos a insertar (repetimos este paso para cada imagen que queramos insertar)



Una vez tenemos todas las imágenes a insertar en la lista, hacemos clic en **Convertir imagen** para añadirlas el Sistema.



Se recomienda usar un formato de imágenes con transparencia (como por ejemplo .png) aunque no es estrictamente necesario.

Editando parámetros del juego

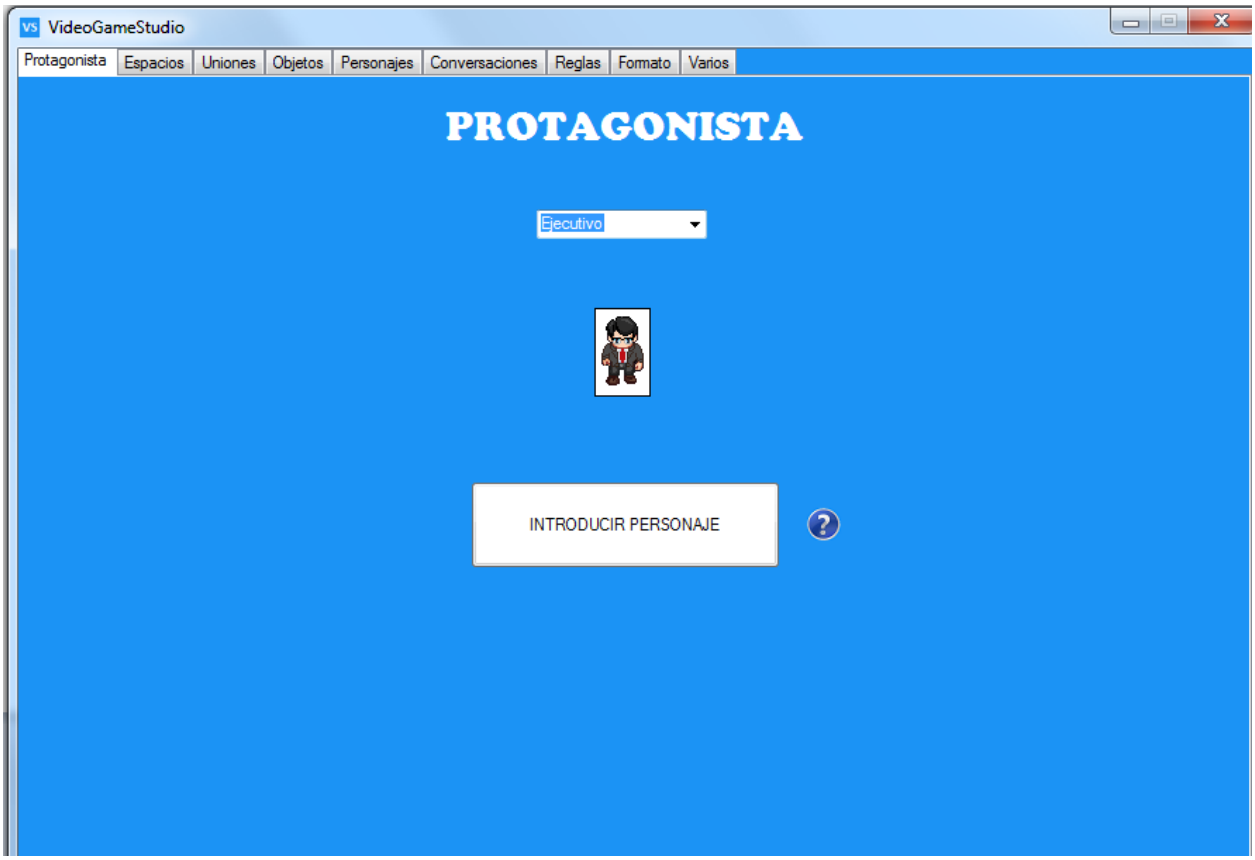
Para comenzar a personalizar los diferentes aspectos de nuestro juego, hacemos clic en el botón **EDITAR JUEGO**

Se nos abrirá el formulario desde donde podemos modificar diferentes parámetros del juego, en dicho formulario se encuentran las siguientes pestañas:

- Protagonista
- Espacios
- Uniones
- Objetos
- Personajes
- Conversaciones
- Reglas
- Varios

Protagonista

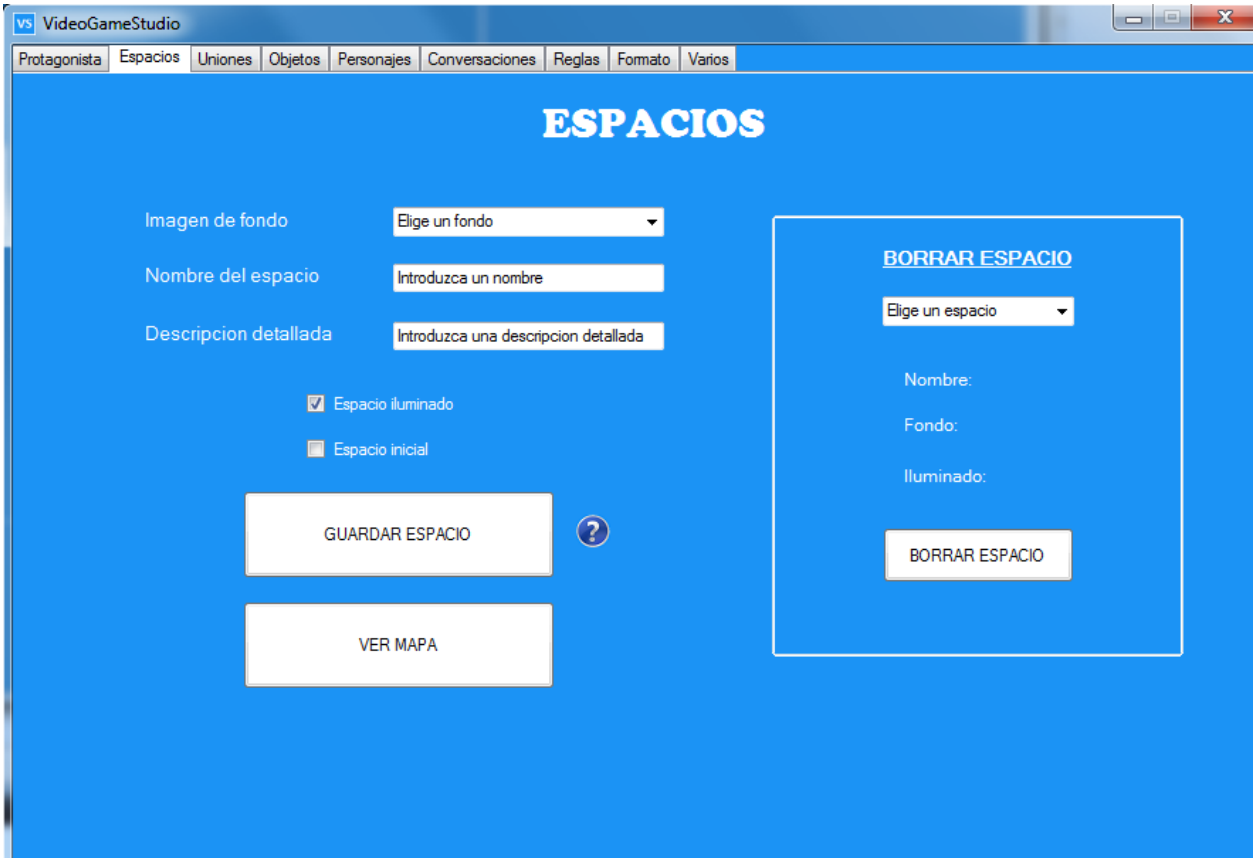
En esta pestaña podemos elegir al personaje que será manejado por el jugador a lo largo del juego.



Al elegir un personaje de la lista se nos muestra una pre visualización del modelo de dicho personaje.
Una vez elegido el personaje deseado, hacemos clic en **INTRODUCIR PERSONAJE** y lo introducimos al sistema.

Espacios

En esta pestaña podemos modificar los diferentes aspectos de los mapas que componen el juego.



Introducir Espacio

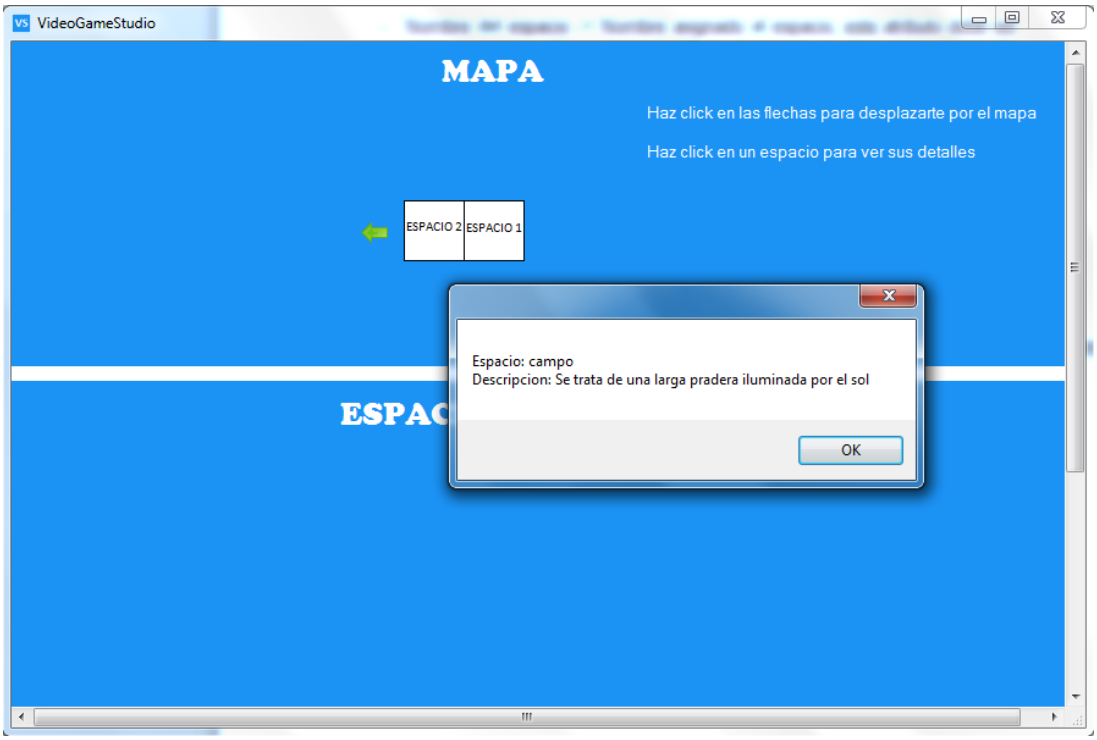
A la hora de introducir un espacio en el Sistema, se permite configurar los siguientes parámetros:

- Imagen de fondo -> Nombre de la imagen de fondo del espacio
- Nombre del espacio -> Nombre asignado al espacio, este atributo debe ser único ya que se utilizara para identificar al espacio en otros pasos del diseño del videojuego (por ejemplo a la hora de crear una unión entre dos espacios).
- Descripción detallada -> Descripción que se mostrara al usuario cuando este haga clic en la opción “Examinar espacio”.

Una vez introducidos los parámetros deseados, hacemos clic en **GUARDAR ESPACIO**

Ver mapa

Permite ver una representación de los espacios y uniones introducidos hasta el momento en el sistema.
Para desplazarnos por el mapa hacemos clic en las flechas, si queremos ver los detalles de un espacio hacemos clic en el espacio correspondiente.

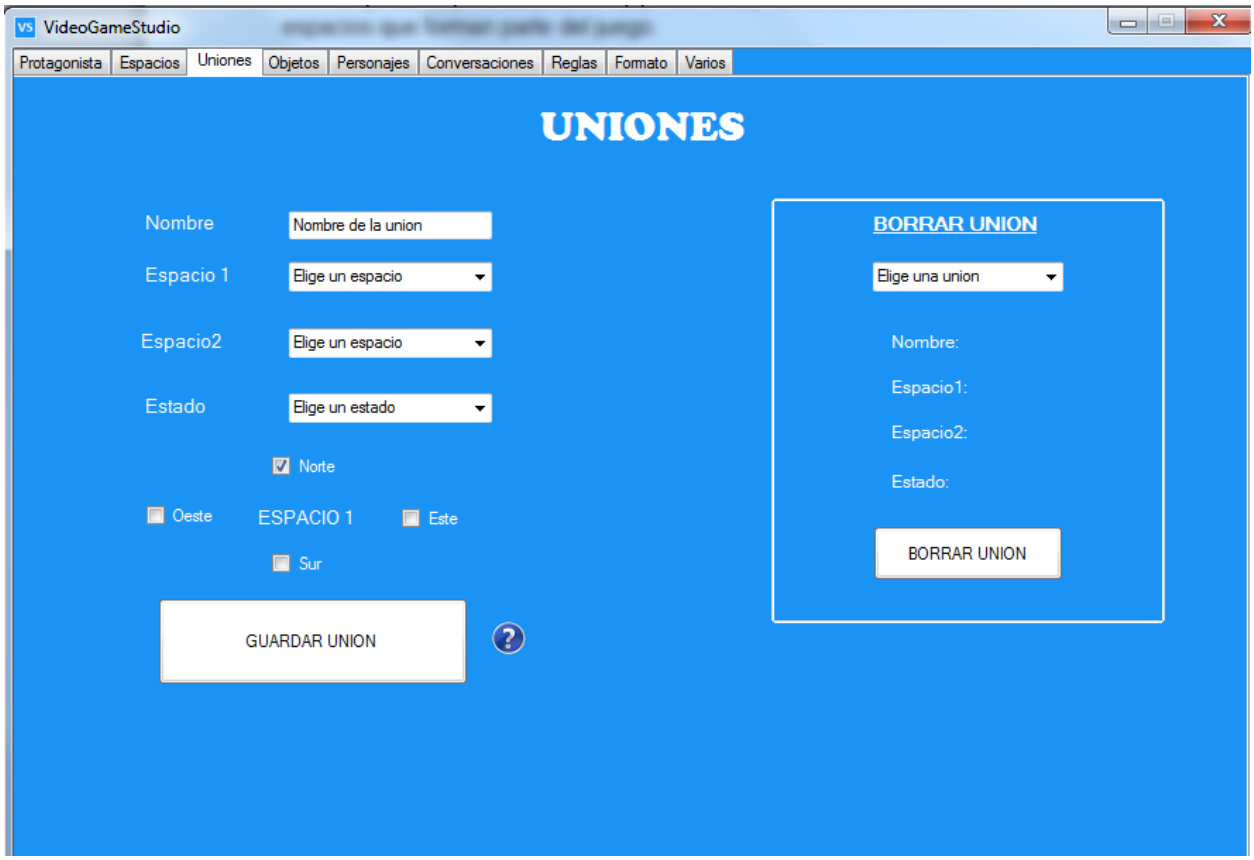


Borrar Espacio

Para eliminar un espacio del Sistema, lo elegimos en la lista (al elegir un espacio se muestran sus detalles).
Una vez elegido el espacio que se quiere borrar, hacemos clic en **BORRAR ESPACIO**

Uniones

En esta pestaña podemos definir y personalizar las conexiones entre los diferentes espacios que forman parte del juego.



Introducir unión

A la hora de introducir una unión en el Sistema, se permite configurar los siguientes parámetros:

- Nombre -> Nombre que identificara a la unión en el sistema, este atributo debe ser único ya que se utilizara para identificar la unión en otros pasos del diseño del videojuego (por ejemplo a la hora de crear una regla que abra la unión).

- Espacio1 -> Nombre del primer espacio conectado por la unión.

- Espacio 2 -> Nombre del segundo espacio conectado por la unión.

- Estado -> Se trata de un parámetro que puede coger dos valores: Abierto (se permite al jugado pasar por la unión) o Cerrado (el usuario no puede pasar)

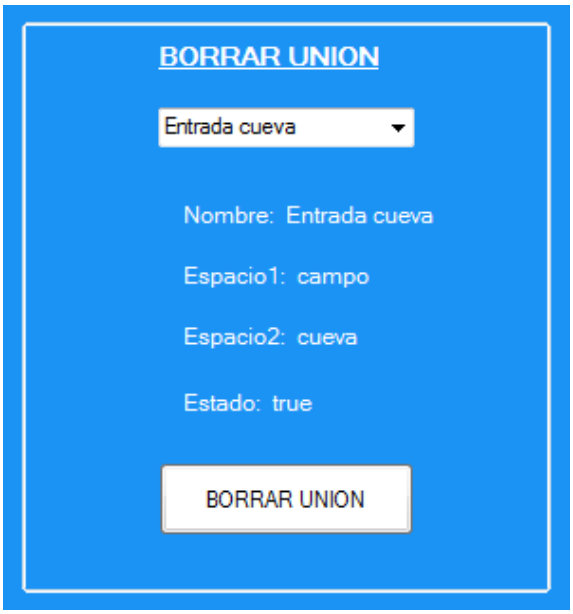
- Posición de la unión -> Indica en que posición, respecto al espacio 1, se encuentra la unión, por ejemplo la siguiente captura muestra un

ejemplo en el que se crea una unión entre la izquierda de campo (Oeste) y la derecha de cueva (Este):

Una vez introducidos los parámetros deseados, hacemos clic en **GUARDAR UNION**

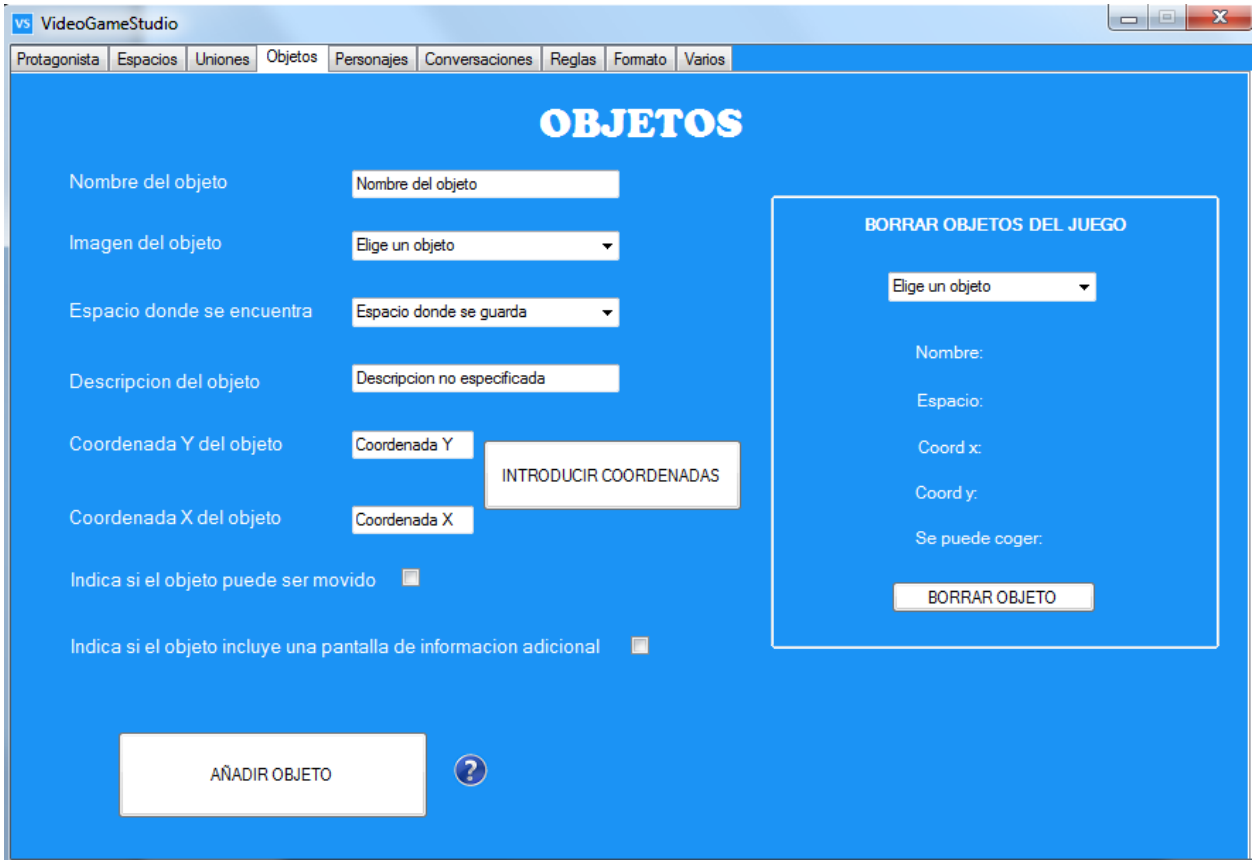
Borrar unión

Para eliminar una unión del Sistema, la elegimos en la lista (al elegir una unión se muestran sus detalles).
Una vez elegida la unión que se quiere borrar, hacemos clic en **BORRAR UNION**



Objetos

En esta pestaña podemos añadir y personalizar objetos al juego.



Introducir objeto

A la hora de introducir un objeto en el Sistema, se permite configurar los siguientes parámetros:

- Nombre del objeto -> Nombre que identificara al objeto en el sistema, este atributo debe ser único ya que se utilizara para identificar el objeto en otros pasos del diseño del videojuego (por ejemplo a la hora de crear una regla que del objeto al usuario).

- Imagen del objeto -> Nombre de la imagen correspondiente al objeto.

- Espacio donde se encuentra -> Nombre del espacio donde se encuentra el objeto. Puede ser tanto un espacio del sistema, como el inventario del jugador.

- Descripción del objeto -> Se trata de la descripción que obtendrá el usuario cuando examine el objeto.

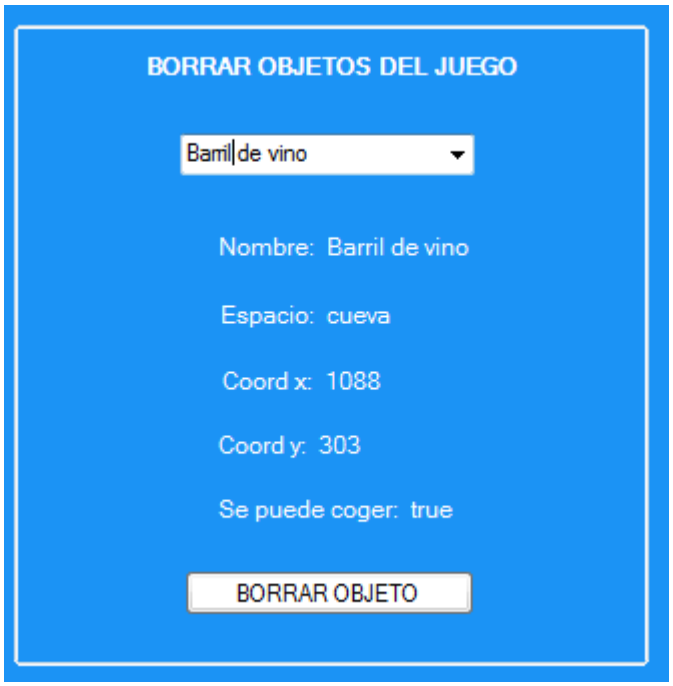
- Coordenadas -> Este parámetro solo se muestra si se ha elegido un espacio distinto al inventario. Para elegir unas coordenadas hacemos

clic en **INTRODUCIR COORDENADAS** y hacemos clic en el lugar del mapa donde queremos colocar el objeto.

Una vez introducidos los parámetros deseados, hacemos clic en **AÑADIR OBJETO**

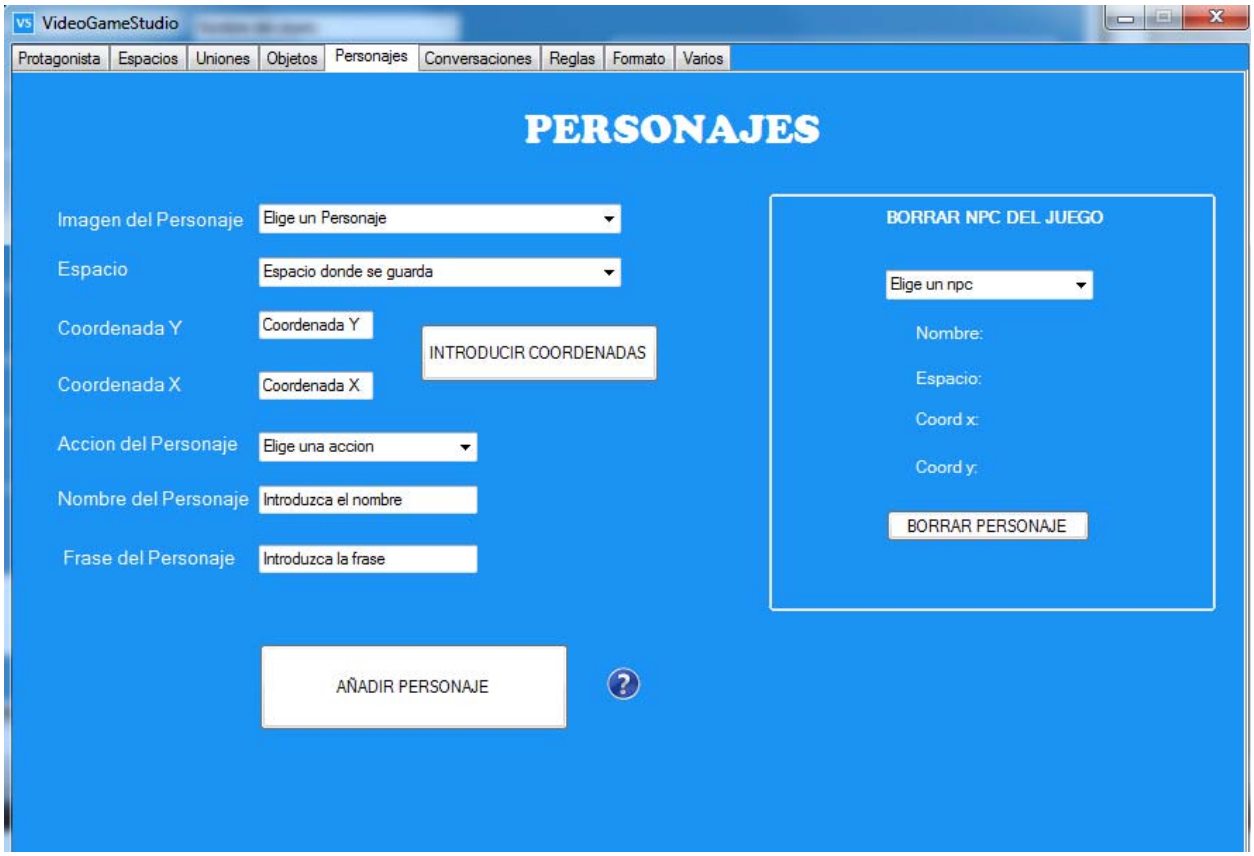
Borrar objeto

Para eliminar un objeto del Sistema, lo elegimos en la lista (al elegir un objeto se muestran sus detalles).
Una vez elegido el objeto que se quiere borrar, hacemos clic en **BORRAR OBJETO**



Personajes

En esta pestaña podemos añadir y personalizar diferentes personajes no controlados por el usuario al juego.



Introducir Personaje

A la hora de introducir un NPC en el Sistema, se permite configurar los siguientes parámetros:

- Imagen del NPC -> Nombre de la imagen correspondiente al NPC.
- Acción del NPC -> Acción que realizara el personaje cuando el usuario interactué con él, esta acción puede ser o bien una de las acciones predefinidas o bien una conversación creada anteriormente.
- Modificador de acción -> Básicamente se trata del parámetro que complementa la acción, por ejemplo en el caso de que se elija la acción “Abrir unión”, modificador será el nombre de la unión a abrir.
- Espacio del NPC -> Nombre del espacio en el que se encuentra el NPC

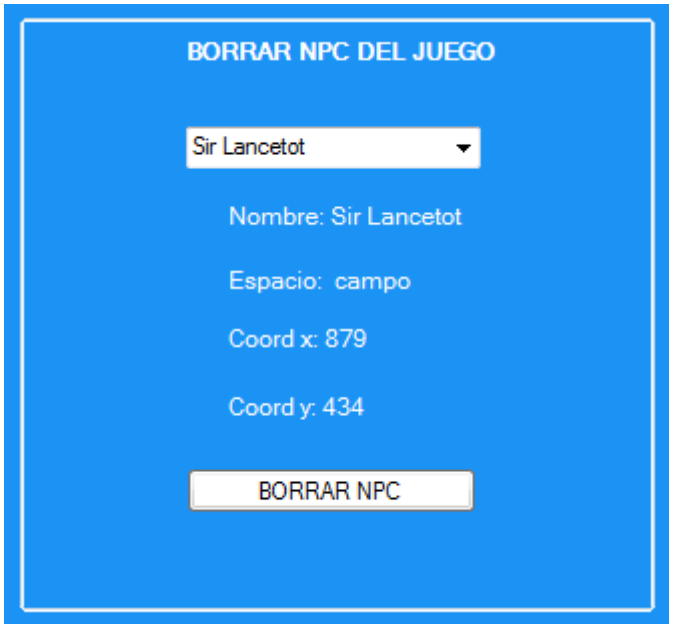
- Nombre del NPC -> Nombre que identificara al NPC en el sistema, este atributo debe ser único ya que se utilizara para identificar al NPC.

- Coordenadas -> Coordenadas donde se encuentra el NPC, para introducir las coordenadas hacemos clic en **INTRODUCIR COORDENADAS** y hacemos clic en el lugar del mapa donde queremos colocar el NPC

Una vez introducidos los parámetros deseados, hacemos clic en **AÑADIR NPC**

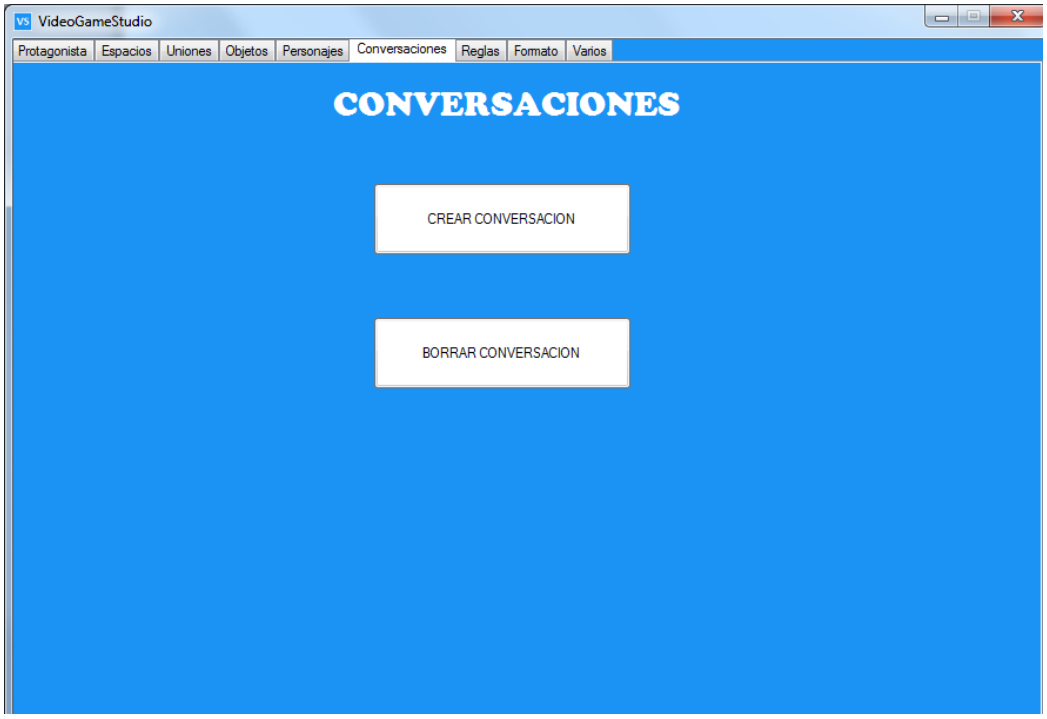
Borrar NPC

Para eliminar un NPC del Sistema, lo elegimos en la lista (al elegir un NPC se muestran sus detalles).
Una vez elegido el NPC que se quiere borrar, hacemos clic en **BORRAR NPC**



Conversaciones

En esta pestaña podemos gestionar todo lo referente a las conversaciones del Sistema, en primer lugar se nos ofrecen dos alternativas (Crear conversación o Borrar conversación).



Crear conversación

Al hacer clic en el botón **CREAR CONVERSACION** se nos abre el siguiente formulario

En este paso del proceso de creación de conversaciones se nos piden los siguientes parámetros:

- Nombre -> Nombre con el que se identificara la conversación en el sistema, este nombre debe ser único para cada conversación.

- Numero de pasos-> Numero de estados que forman la conversación.

Una vez introducido el valor de estos parámetros, hacemos clic en **SIGUIENTE** abriéndose el siguiente formulario:

En este formulario se permite personalizar los diferentes parámetros de cada paso de la conversación, dichos parámetros son los siguientes:

- Frase inicial -> Frase que dice el NPC al acercarse el usuario sin decir nada.

- Respuesta esperada -> Frase de respuesta que espera recibir el NPC del usuario.

- Frase si respuesta incorrecta -> Frase que pronuncia el NPC si el usuario introduce una respuesta diferente a la respuesta esperada, también es posible seleccionar “No decir nada”

- Acción si respuesta incorrecta -> Acción que realiza el NPC si el usuario introduce una respuesta diferente a la respuesta esperada

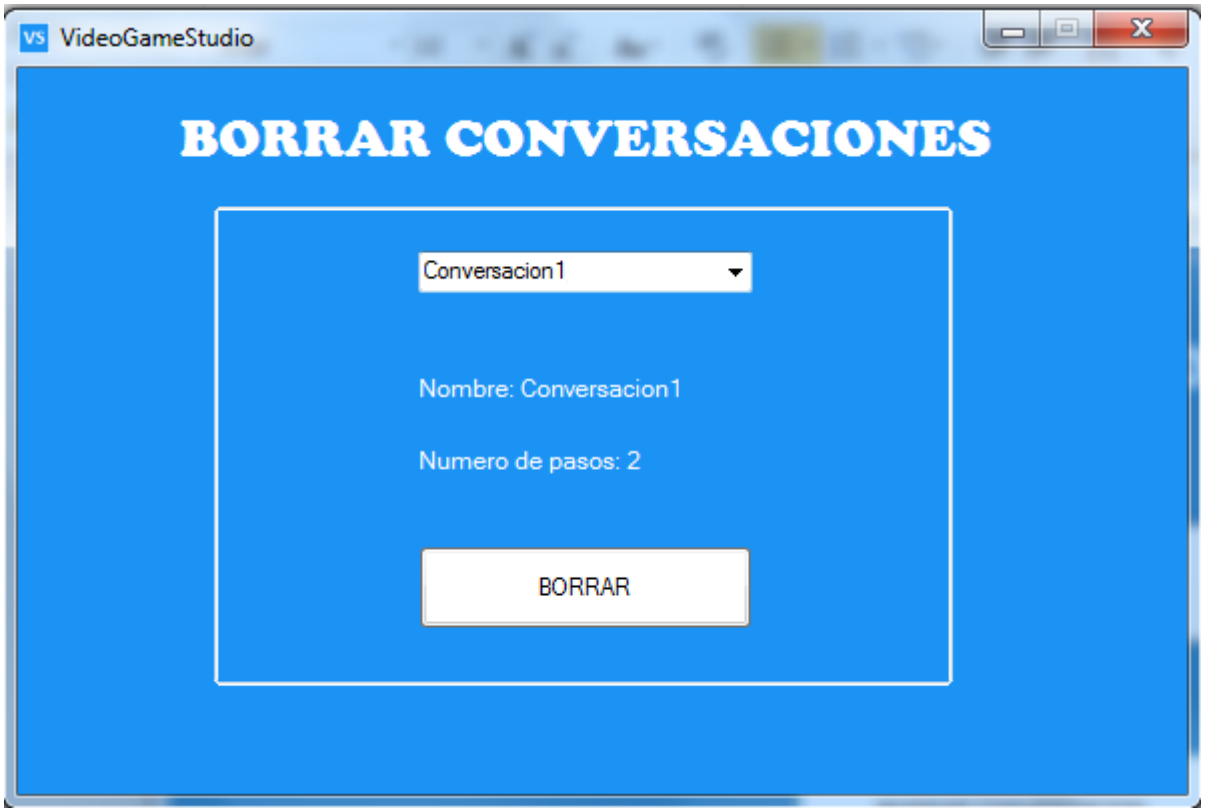
- Frase si respuesta correcta -> Frase que pronuncia el NPC si el usuario introduce la respuesta esperada, también es posible seleccionar “No decir nada”

- Acción si respuesta correcta -> Acción que realiza el NPC si el usuario introduce la respuesta esperada

Una vez rellenados los parámetros de todos los pasos de la conversación, hacemos clic en **CREAR CONVERSACION** para introducir la conversación en el sistema.

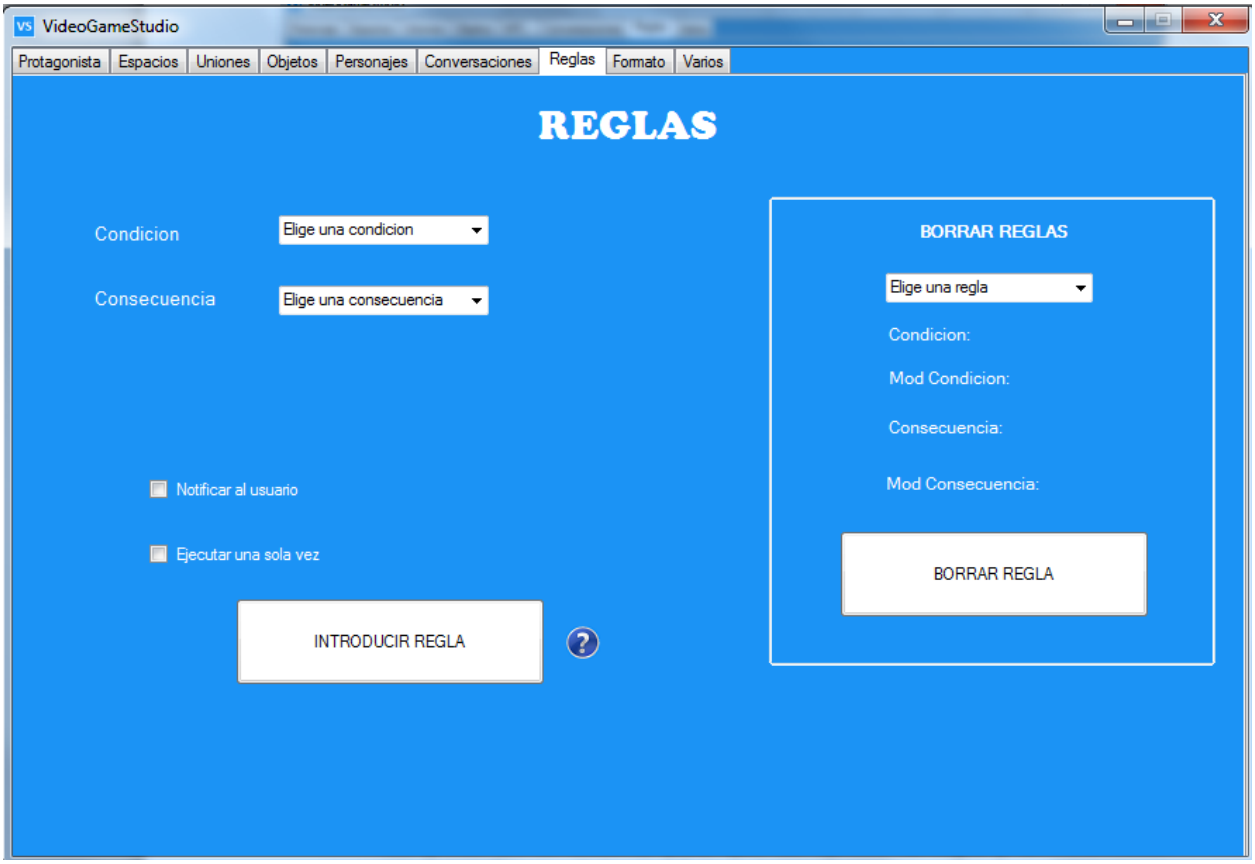
Borrar conversación

Para eliminar una conversación del Sistema, la elegimos en la lista (al elegir una conversación se muestran sus detalles).
Una vez elegida la conversación que se quiere borrar, hacemos clic en **BORRAR**



Reglas

En esta pestaña se puede gestionar todo lo referente a reglas.
Una regla básicamente consiste en realizar una acción cuando se cumpla una cierta condición en el Sistema, por ejemplo iluminar un determinado espacio cuando el usuario recoja una linterna.



Introducir Regla

A la hora de introducir una regla en el Sistema, se permite configurar los siguientes parámetros:

- Condición de la regla y su modificador -> Se trata de la condición que debe cumplirse en el Sistema para que se ejecute la regla y el modificador correspondiente a dicha condición.

- Consecuencia de la regla y su modificador -> Se trata de la acción que realiza la regla cuando se cumple la condición y el modificador correspondiente a esa acción.

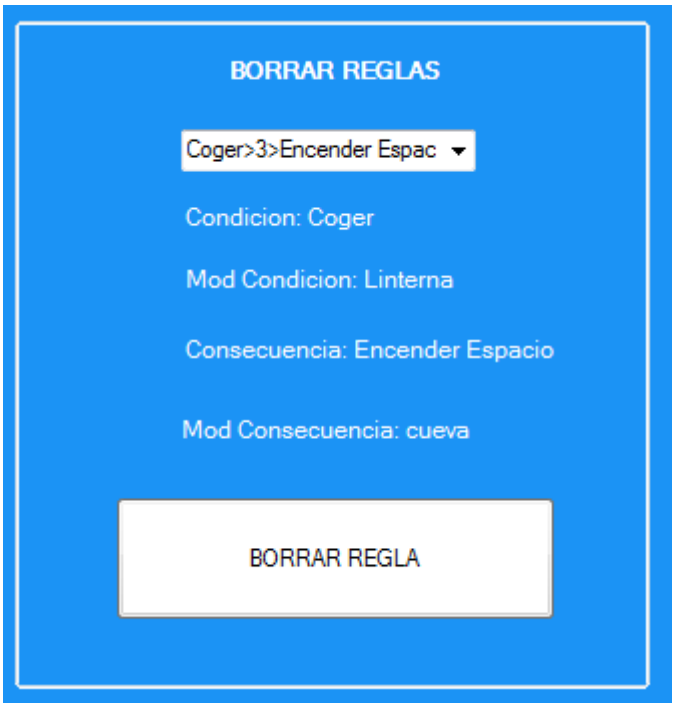
- Notificar al usuario -> Indica si se debe notificar al usuario de la ejecución de la regla o si la regla debe ejecutarse a espaldas del jugador.

- Ejecutar una sola vez -> Indica si la regla debe ejecutarse cada vez que se cumpla la condición o si solo se ejecuta la primera vez que se cumple la condición.

Una vez introducidos los parámetros, hacemos clic en **INTRODUCIR REGLA** y guardamos la regla en el Sistema.

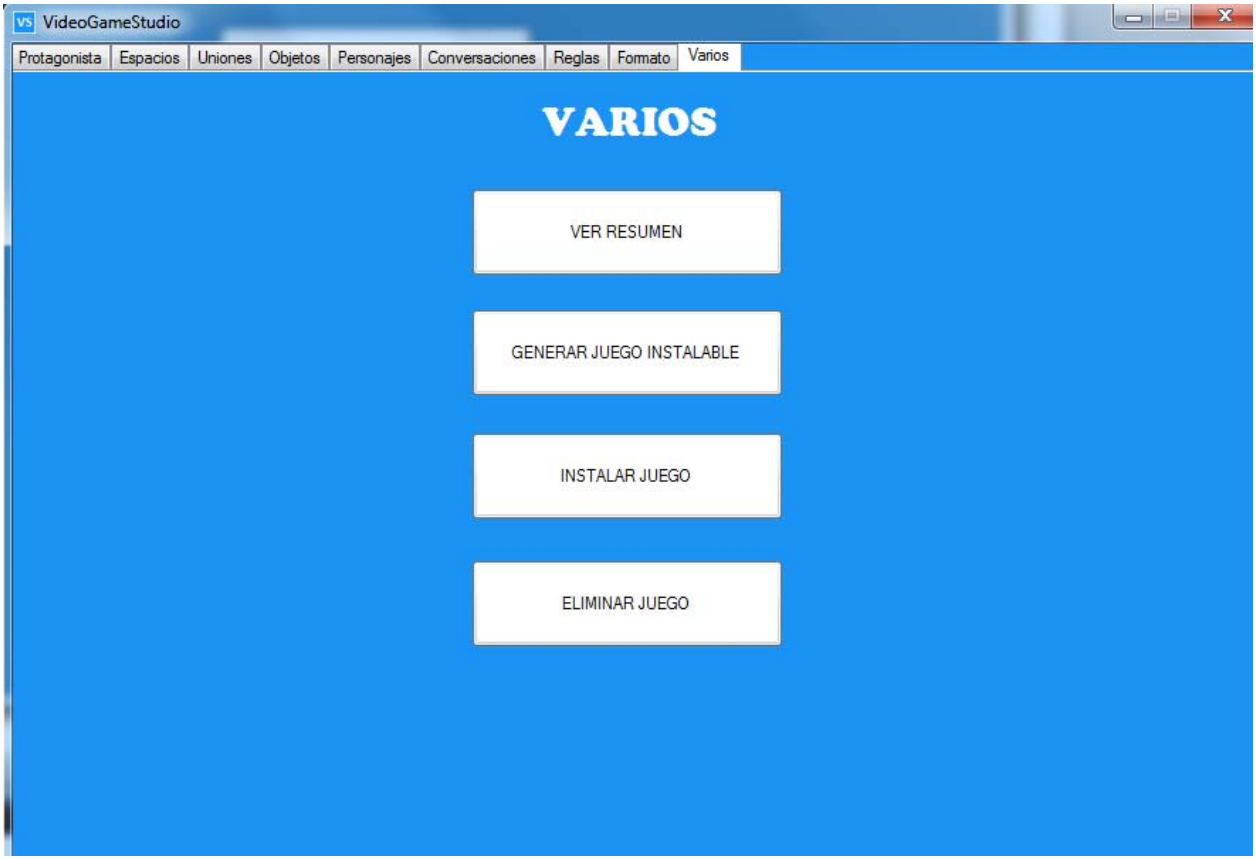
Borrar regla

Para eliminar una regla del Sistema, la elegimos en la lista (al elegir una regla se muestran sus detalles).
Una vez elegida la regla que se quiere borrar, hacemos clic en **BORRAR**



Varios

En esta pestaña se muestran las opciones que no tienen cabida en las demás pestañas.



La opción **VER RESUMEN** permite ver todos los progresos realizados en el desarrollo del juego hasta el momento.

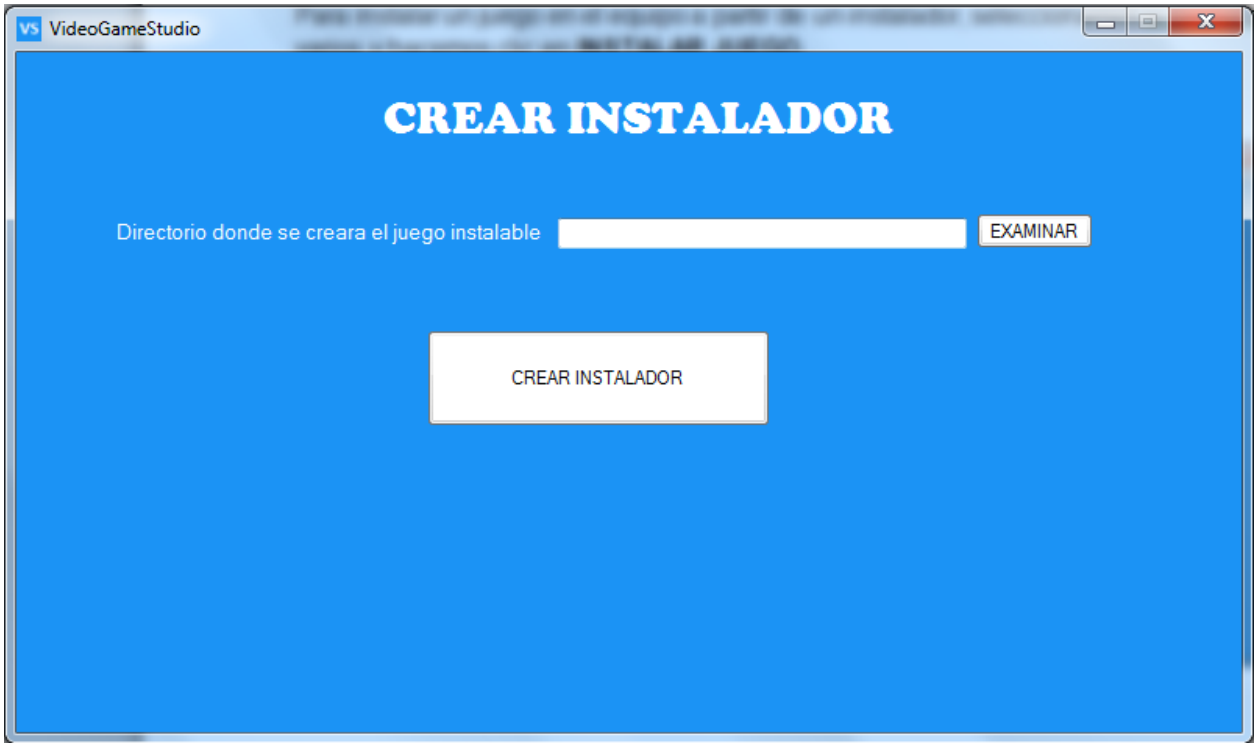
La opción **GENERAR JUEGO INSTALABLE** permite crear un instalador para el juego que hemos creado (tratado en detalle en la página 25).

La opción **INSTALAR JUEGO** permite instalar un juego a partir de un instalador creado previamente (tratado en detalle en la página 26).

La opción **ELIMINAR JUEGO** borra todos los datos del juego.

Crear instalador de un juego

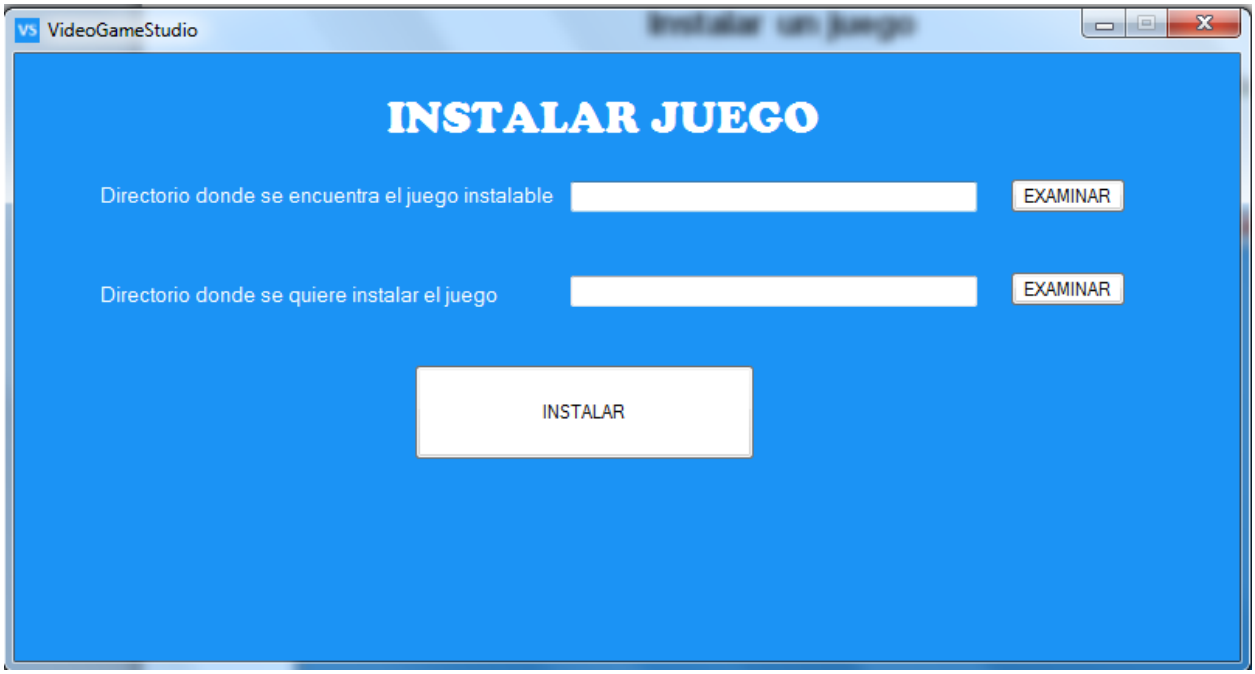
Para crear un instalador para nuestro juego, seleccionamos la pestaña varios y hacemos clic en **GENERAR JUEGO INSTALABLE.**
Se nos abrirá el siguiente formulario:



Simplemente elegimos un directorio valido, hacemos clic en **CREAR INSTALADOR** y se generaran en dicho directorio los archivos necesarios para instalar el juego en otro equipo.

Instalar un juego

Para instalar un juego en el equipo a partir de un instalador, seleccionamos la pestaña varios y hacemos clic en **INSTALAR JUEGO.**
Se nos abrirá el siguiente formulario:



Se nos pide introducir dos rutas:

- La primera de ellas corresponde a la ruta donde se encuentra el instalador del juego.

- La segunda ruta corresponde al directorio donde se quiere instalar el juego

Una vez introducidas las dos rutas, hacemos clic en **INSTALAR** y se instalara el juego en la ruta especificada.

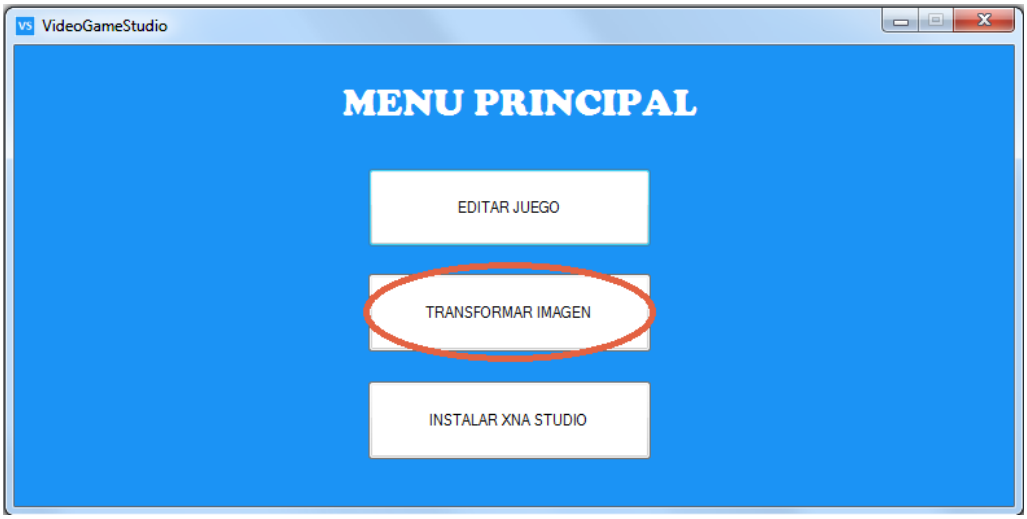
Creación de un juego pasó a paso

Resumen

A lo largo de este tutorial se va a ir desarrollando un pequeño juego en el que se intentara utilizar la mayoría de las características que ofrece la aplicación.

Primer paso: Introduciendo las imágenes en el sistema

En primer lugar vamos a introducir en el sistema, las imágenes de los objetos, personajes y espacios que necesitamos para crear nuestro juego.
Para ello seleccionamos la opción “Transformar imagen” en el menú principal



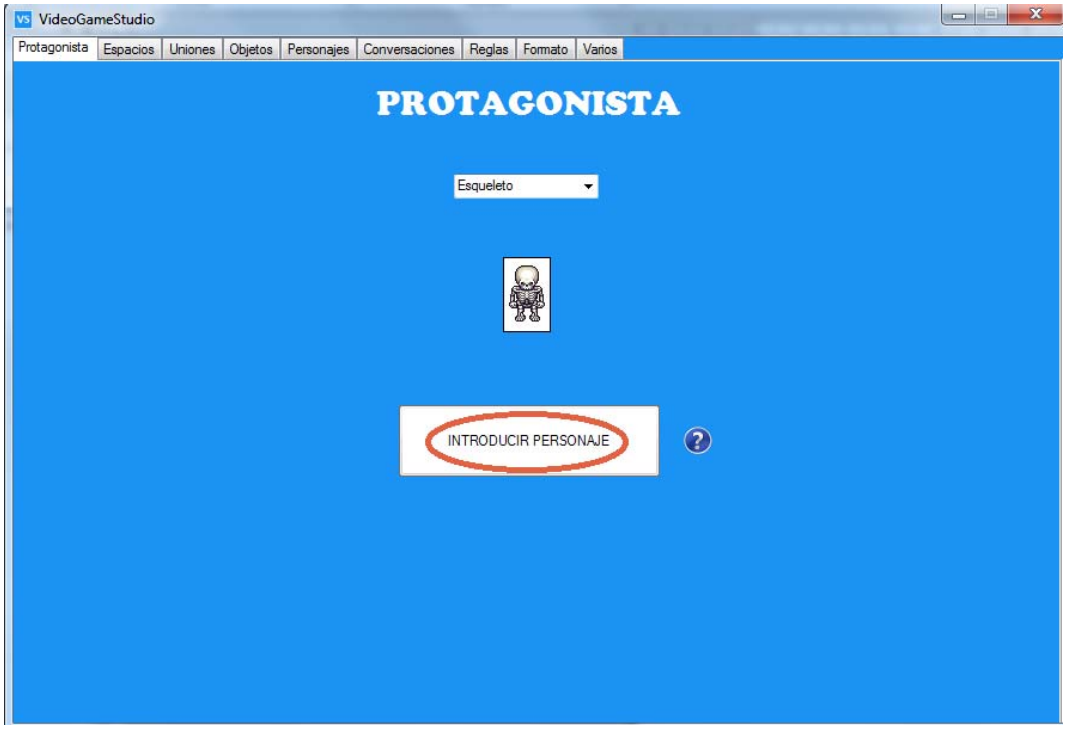
Una vez seleccionada dicha opción se nos abrirá la pantalla desde la que podemos introducir imágenes en el sistema. Seleccionamos el tipo de objeto al que corresponde la imagen a insertar, metemos las imágenes en la lista y hacemos clic en “Convertir imagen”.

Añadiremos al sistema las siguientes imágenes con sus respectivos tipos:
- Espada – Objeto
- Antorcha – Objeto
- Cueva – Fondo
- Soldado – NPC
- Capitán – NPC

Una vez introducidas las imágenes, volvemos al menú principal y hacemos clic en “EDITAR JUEGO” para pasar al siguiente paso del diseño de nuestro videojuego.

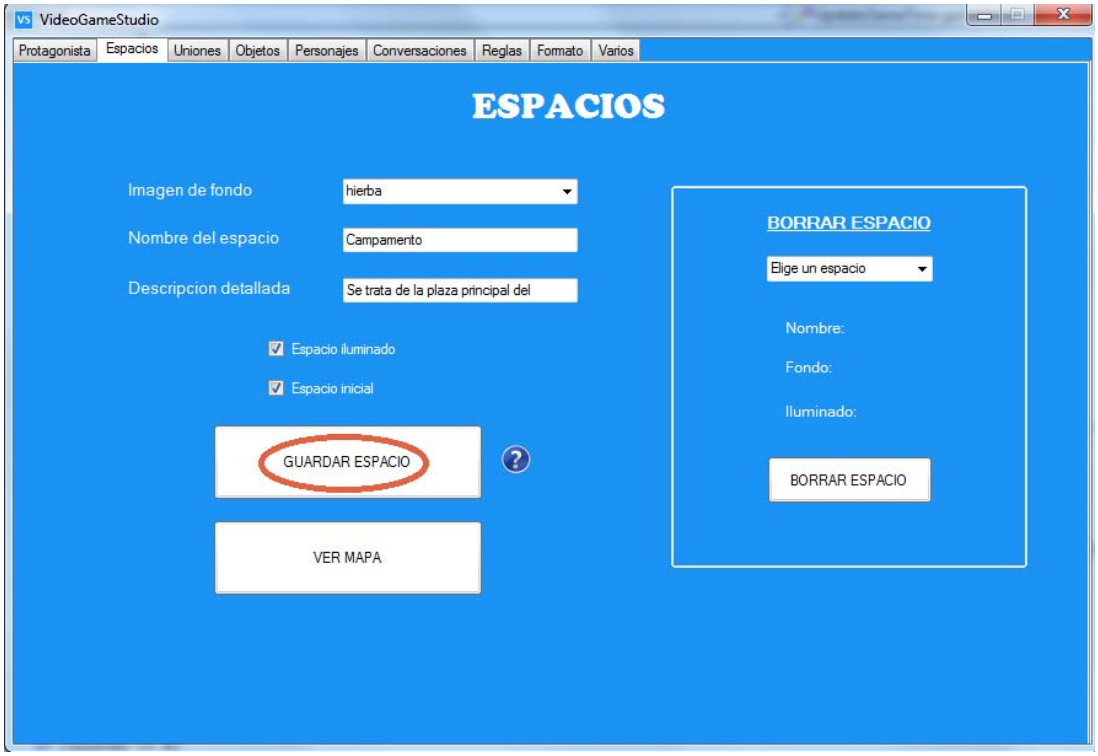
Segundo paso: Eligiendo al protagonista de nuestro juego

En la pestaña de personajes, elegimos el modelo del personaje que protagonizara nuestro juego, en este caso seleccionamos el modelo “Esqueleto” y hacemos clic en INTRODUCIR PERSONAJE.



Tercer Paso: Creando los niveles de nuestro juego

En la pestaña de espacios, rellenamos los diferentes campos para la creación de los niveles de nuestro juego, cuando hayamos rellenado los campos correctamente, hacemos clic en GUARDAR ESPACIO.



Concretamente crearemos cuatro espacios con los siguientes datos:

- Espacio1
  - Imagen - Hierba
  - Nombre - Campamento
  - Descripción – Se trata de la plaza principal del campamento
  - Espacio iluminado - Si
  - Espacio inicial – Si

- Espacio2
  - Imagen - Cueva
  - Nombre - Cueva
  - Descripción – Está todo muy oscuro pero en un rincón puedes observar una espada tirada en el suelo
  - Espacio iluminado - No

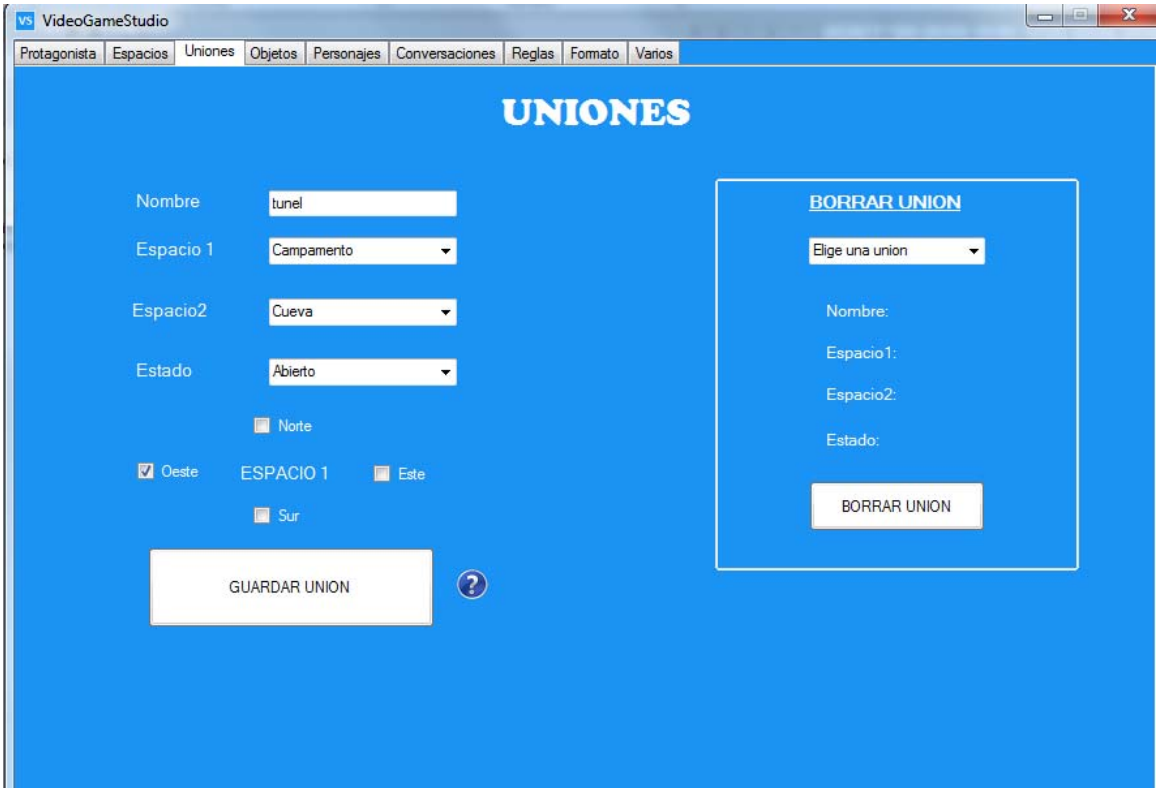
- Espacio inicial - No

- Espacio3
  - Imagen – Hierba
  - Nombre - Norte del campamento
  - Descripción – Se trata de la zona norte del campamento
  - Espacio iluminado - Si
  - Espacio inicial - No

- Espacio4
  - Imagen – Nieve
  - Nombre - Bosque
  - Descripción – Te encuentras en un bosque al norte del campamento
  - Espacio iluminado - Si
  - Espacio inicial – No

Cuarto Paso: Creando las uniones entre espacios de nuestro juego

Para ello seleccionamos la pestaña Uniones y creamos las diferentes conexiones entre espacios de nuestro juego



Concretamente creamos las siguientes uniones:

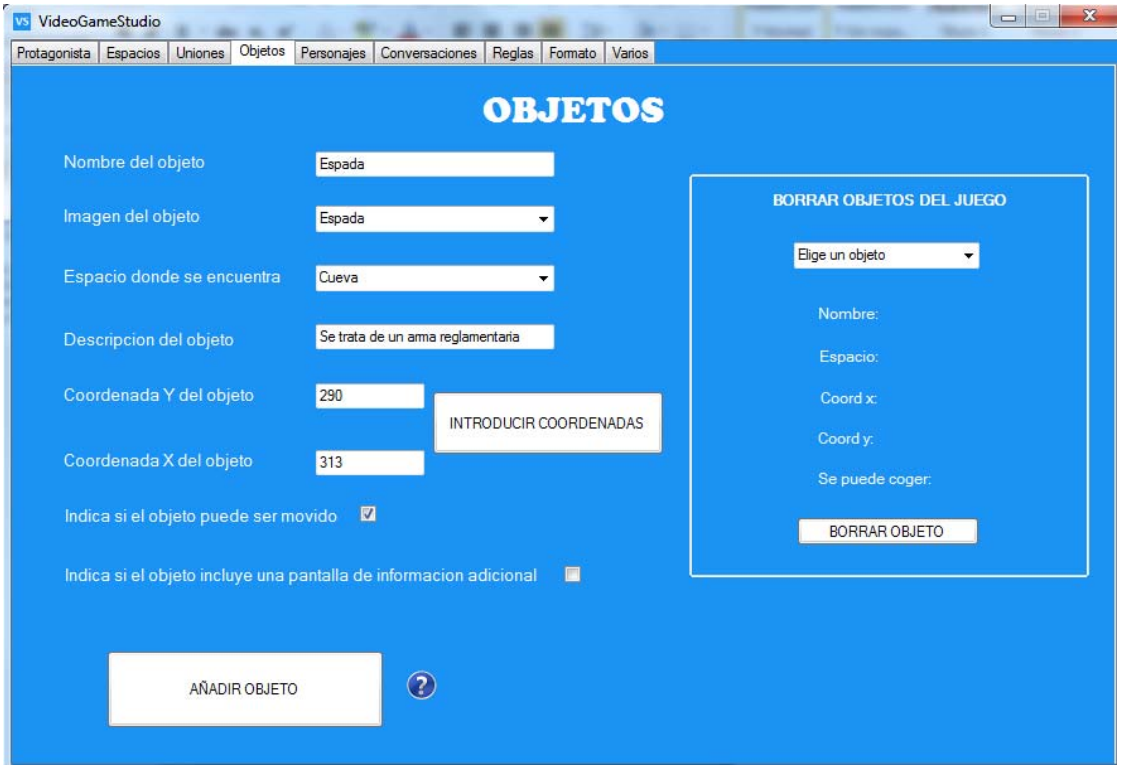
- Union1
  - Nombre – Túnel
  - Espacio1 – Campamento
  - Espacio2 - Cueva
  - Estado – Abierto
  - Posición – Oeste

- Union2
  - Nombre – Camino-Norte
  - Espacio1 – Campamento
  - Espacio2 - Norte del campamento
  - Estado – Cerrado
  - Posición - Norte

- Union3
  - Nombre – Camino-Bosque
  - Espacio1 – Norte del Campamento
  - Espacio2 - Bosque
  - Estado – Cerrado
  - Posición – Norte

Quinto Paso: Creando los objetos del juego

Para ello seleccionamos la pestaña Objetos y creamos los diferentes objetos que formaran parte de nuestro juego.



Se crearan 3 objetos para este juego de prueba con los siguientes atributos
- Objeto1

- Nombre - Espada
- Imagen - Espada
- Espacio donde se encuentra - Cueva
- Descripción – Se trata de un arma reglamentaria.
- Se puede mover? – Si
- Incluye información adicional - No

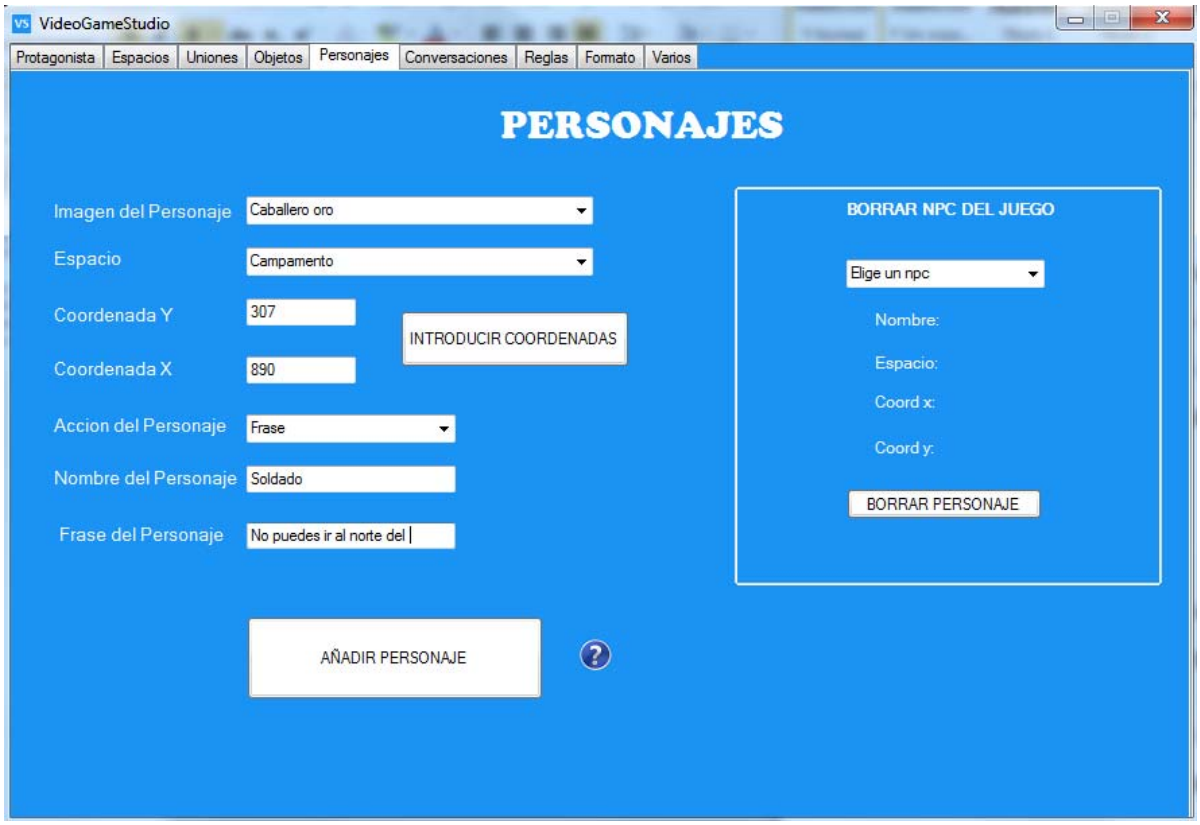
- Objeto2

- Nombre - Antorcha
- Imagen - Antorcha
- Espacio donde se encuentra - Campamento
- Descripción – Se trata de una antorcha, puedo usarla para explorar la cueva.
- Se puede mover? – Si
- Incluye información adicional – No

- Objeto3
  - Nombre - Mapa
  - Imagen - Mapa
  - Espacio donde se encuentra - Ninguno
  - Descripción – Ahora que tengo el mapa ya puedo salir a explorar el bosque sin peligro.
  - Se puede mover? – Si
  - Incluye información adicional - No

Sexto Paso: Creando los personajes del juego

Para ello seleccionamos la pestaña Personajes y personalizamos los diferentes personajes que formaran parte de nuestro juego



Concretamente crearemos dos personajes:

- Personaje1
  - Imagen - Soldado
  - Espacio - Campamento

- Acción - Frase
- Nombre - Soldado
- Frase – No puedes ir al norte del campamento sin tu arma, es demasiado peligroso.

- Personaje2

- Imagen - Capitan
- Espacio – Norte del campamento
- Acción - ConversacionCapitan
- Nombre – Capitan

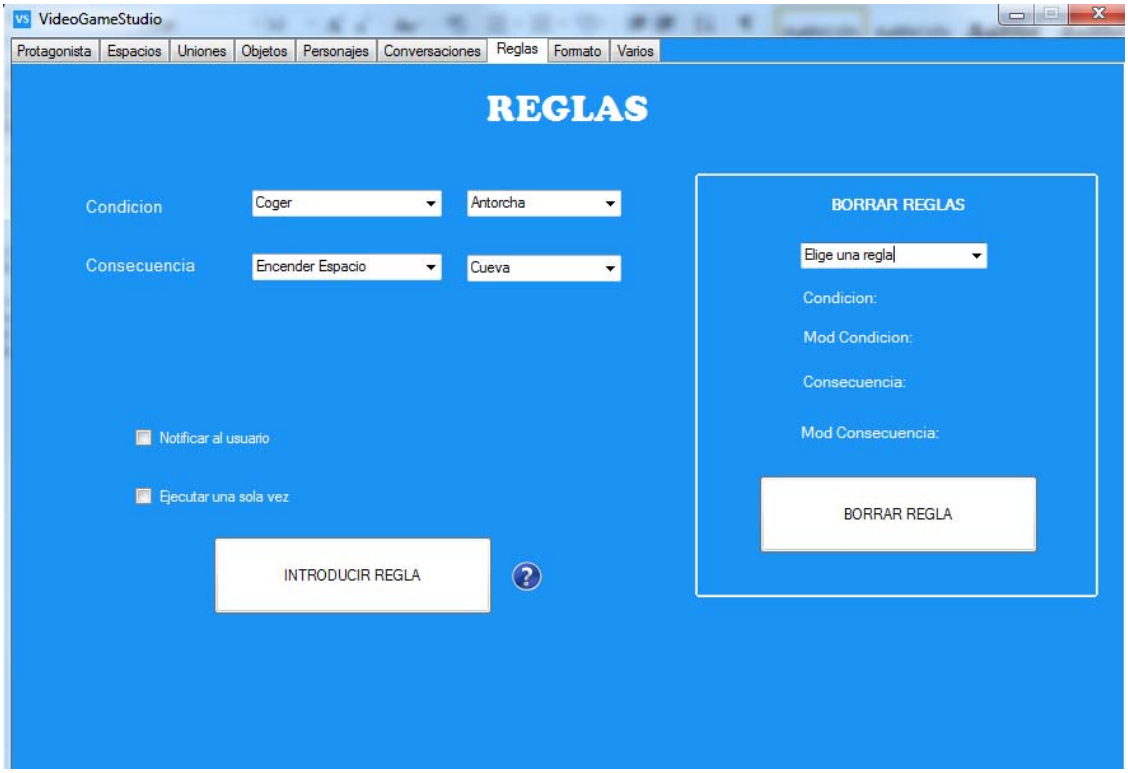
Séptimo Paso: Creando las conversaciones del juego

Para crear la conversación seleccionamos la pestaña conversaciones y hacemos clic en la opción “CREAR CONVERSACION”.
En la pantalla correspondiente al primer paso de la creación de conversaciones, introducimos el nombre de la conversación, el número de pasos de esta y hacemos clic en “SIGUIENTE”:

En la siguiente pantalla editamos el paso de la conversación para que tenga el comportamiento esperado y hacemos clic en CREAR CONVERSACION.

Octavo Paso: Creando las reglas del juego

Seleccionamos la pestaña de reglas y creamos las reglas necesarias para el correcto funcionamiento del juego



Concretamente crearemos 4 reglas diferentes:

- Regla1
  - Condición – Coger Antorcha
  - Consecuencia – Iluminar Cueva
  - Notificación – No

- Regla2
  - Condición – Coger Espada
  - Consecuencia – Abrir Camino-Norte
  - Notificación - No

- Regla3
  - Condición – Coger Mapa
  - Consecuencia – Abrir Camino-Bosque
  - Notificación - No

- Regla4
  - Condición – Entrar Bosque
  - Consecuencia – Fin de juego
  - Notificación – Si
  - Mensaje – Enhorabuena, has completado el juego

Una vez introducidas estas reglas hemos terminado de crear nuestro primer juego de manera satisfactoria.

Información adicional

En este apartado del manual se incluirá información externa a la aplicación pero que pueden resultar útil para realizar un videojuego.

Transparencias en PowerPoint

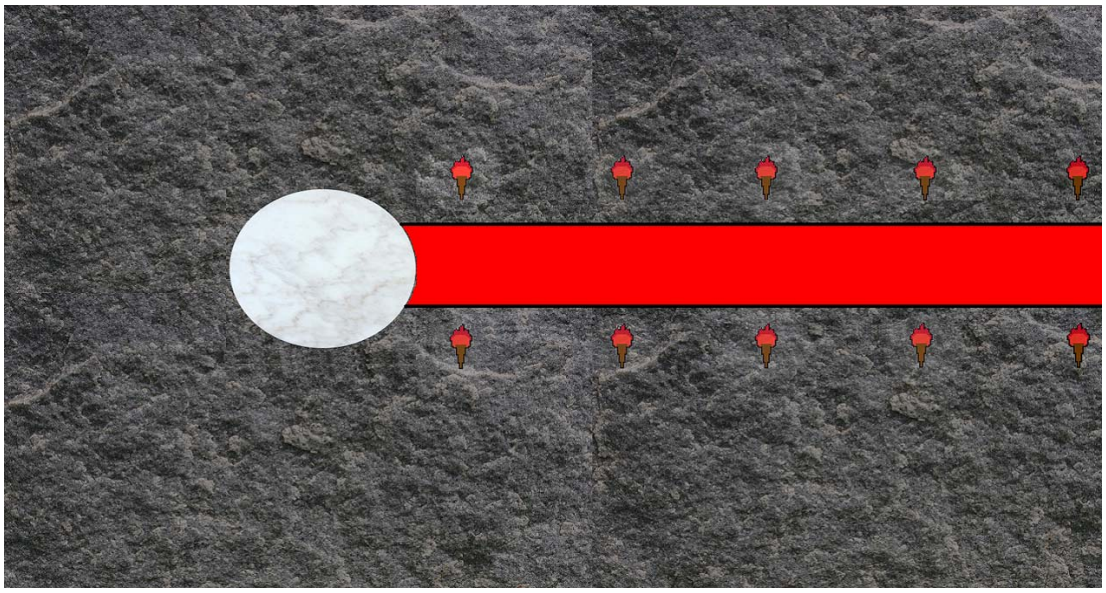
Existen diversas editores capaces de crear transparencia en imágenes, aquí se va a explicar cómo realizar este proceso en el programa PowerPoint por tratarse de una herramienta de fácil acceso.

Para quitar el fondo de una imagen y ponerlo transparente en PowerPoint, insertamos la imagen y hacemos clic en la opción “Quitar Fondo”, después para guardar la imagen hacemos botón derecho -> Guardar como imagen.

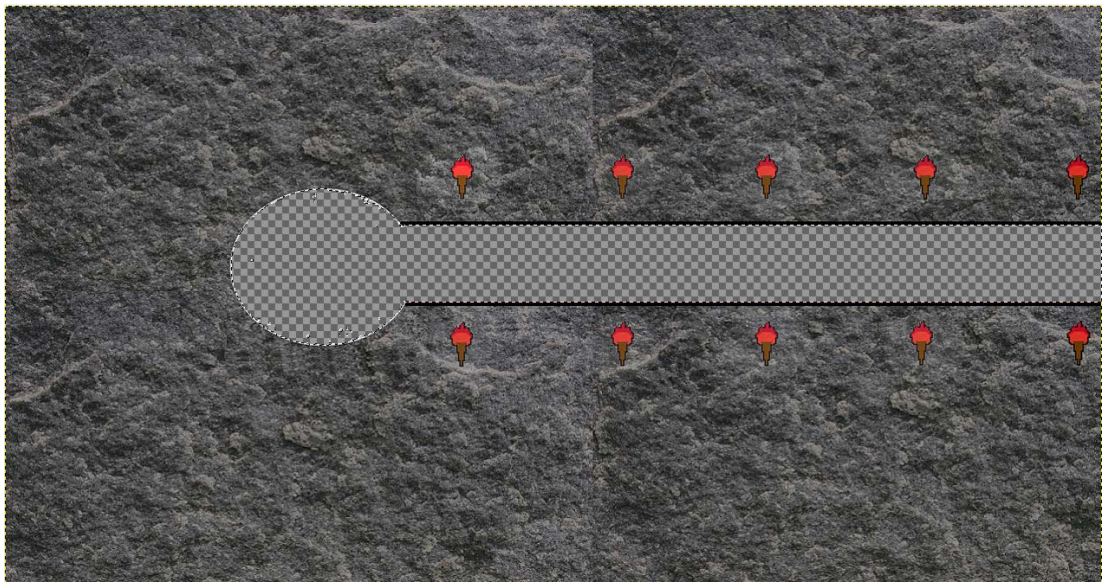
Una vez guardada la imagen, ya estaría lista para su uso en la aplicación.

Como crear zonas impracticables

En muchas ocasiones, dentro de nuestro juego nos interesara que haya zonas en las que el usuario no pueda caminar, para conseguir esto en nuestro juego necesitaremos utilizara un objeto con transparencias.
Partiremos del siguiente fondo en el que queremos que solo se pueda caminar por el pasillo y el círculo central:



Para conseguir esto sería necesario crear un objeto que contuviera transparencias de la siguiente forma:



Vista del sistema

En este apartado del manual se explicara brevemente el tipo de vista que se sigue en el juego para facilitar al usuario la comprensión de la misma.

La vista del juego es una vista desde arriba en dos dimensiones pero con una ligera inclinación para dar una sensación de profundidad a los modelos.

Anexo 2: Resultados de las pruebas unitarias realizadas

A continuación se incluyen las diferentes pruebas unitarias que se han realizado para comprobar el correcto funcionamiento del sistema.

Prueba numero 1

Aplicación a probar	Interfaz (Pestaña Personaje)
Descripción de la prueba	Se va a intentar cambiar el personaje principal del juego sin seleccionar ningún personaje de la lista
Resultado esperado	Se espera que la aplicación no realice el cambio e informe al usuario de que es necesario elegir un personaje de la lista
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio

Prueba numero 2

Aplicación a probar	Interfaz (Pestaña Espacios)
Descripción de la prueba	Se va a intentar introducir un espacio sin seleccionar una imagen de fondo para dicho espacio
Resultado esperado	Se espera que la aplicación no introduzca el espacio en el sistema e informe al usuario de que es necesario elegir una imagen para el espacio.
Resultado obtenido	La aplicación ha permitido introducir el espacio sin problema sin avisar al usuario lo que puede llevar a posibles inconsistencias y errores cuando se ejecute el juego.
Cambios realizados	Se ha procedido a controlar este error para que la aplicación no permita introducir un espacio sin seleccionar imagen de fondo

Prueba numero 3

Aplicación a probar	Interfaz (Pestaña Espacios)
Descripción de la prueba	Se va a intentar introducir un espacio con el mismo nombre que un espacio introducido previamente en el sistema
Resultado esperado	Se espera que la aplicación no introduzca el espacio en el sistema e informe al usuario de que es necesario que el nombre del espacio sea único.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio

Prueba numero 4

Aplicación a probar	Interfaz (Pestaña Uniones)
Descripción de la prueba	Se va a intentar introducir una unión sin haber elegido dos espacios a unir
Resultado esperado	Se espera que la aplicación no introduzca la unión en el sistema e informe al usuario de que es necesario introducir dos espacios a unir entre sí.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio

Prueba numero 5

Aplicación a probar	Interfaz (Pestaña Uniones)
Descripción de la prueba	Se va a intentar introducir una unión sin haber especificado si la unión está abierta o cerrada.
Resultado esperado	Se espera que la aplicación no introduzca la unión en el sistema e informe al usuario de que es necesario especificar si la unión está abierta o si la unión está cerrada.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio

Prueba numero 6

Aplicación a probar	Interfaz (Pestaña Uniones)
Descripción de la prueba	Se va a intentar introducir una unión sin haber seleccionado la posición relativa de la unión respecto al primer espacio.
Resultado esperado	Se espera que la aplicación no introduzca la unión en el sistema e informe al usuario de que es necesario especificar la posición relativa de la unión
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio

Prueba numero 7

Aplicación a probar	Interfaz (Pestaña Uniones)
Descripción de la prueba	Se va a intentar introducir una unión que una un espacio consigo mismo
Resultado esperado	Se espera que la aplicación no introduzca la unión en el sistema e informe al usuario de que no es posible unir un espacio a si mismo
Resultado obtenido	La aplicación ha permitido introducir la unión en el sistema sin avisar al usuario del error
Cambios realizados	Se ha controlado el error para que no su puede permitir unir un espacio consigo mismo a partir de una unión.

Prueba numero 8

Aplicación a probar	Interfaz (Pestaña Uniones)
Descripción de la prueba	Se va a intentar introducir una unión con el mismo nombre que una unión introducida previamente en el sistema
Resultado esperado	Se espera que la aplicación no introduzca la unión en el sistema e informe al usuario de que es necesario que el nombre de la unión sea único.
Resultado obtenido	La aplicación ha permitido introducir la unión en el sistema sin avisar del error.
Cambios realizados	Se ha añadido un pequeño fragmento de código que controla que el nombre de la unión sea único para el sistema para evitar que se produzca este error.

Prueba numero 9

Aplicación a probar	Interfaz (Pestaña Objetos)
Descripción de la prueba	Se va a intentar introducir en el sistema un objeto sin haber elegido una imagen para dicho objeto.
Resultado esperado	Se espera que la aplicación no introduzca el objeto en el sistema e informe al usuario de que es necesario elegir la imagen del objeto.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio

Prueba numero 10

Aplicación a probar	Interfaz (Pestaña Objetos)
Descripción de la prueba	Se va a intentar introducir en el sistema un objeto sin haber especificado el espacio en el que se encontrara dicho objeto
Resultado esperado	Se espera que la aplicación no introduzca el objeto en el sistema e informe al usuario de que es necesario elegir el espacio donde se encontrara el objeto.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio

Prueba numero 11

Aplicación a probar	Interfaz (Pestaña Objetos)
Descripción de la prueba	Se va a intentar introducir en el sistema un objeto sin haber especificado las coordenadas del espacio en las que está situado el objeto.
Resultado esperado	Se espera que la aplicación no introduzca el objeto en el sistema e informe al usuario de que es necesario introducir unas coordenadas para dicho objeto
Resultado obtenido	La aplicación ha permitido introducir el objeto en el sistema pese a que a debía haber avisado del error.
Cambios realizados	Se ha añadido una comprobación para las coordenadas del objeto, no solo se comprueba que se hayan insertado, sino que sean correctas.

Prueba numero 12

Aplicación a probar	Interfaz (Pestaña Objetos)
Descripción de la prueba	Se va a intentar introducir en el sistema un objeto en unas coordenadas incorrectas
Resultado esperado	Se espera que la aplicación no introduzca el objeto en el sistema e informe al usuario de que es necesario introducir unas coordenadas correctas para el objeto
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 13

Aplicación a probar	Interfaz (Pestaña Objetos)
Descripción de la prueba	Se va a intentar introducir en el sistema un objeto con el mismo nombre que otro objeto introducido previamente en el sistema
Resultado esperado	Se espera que la aplicación no introduzca el objeto en el sistema e informe al usuario de que el nombre del objeto debe de ser único.
Resultado obtenido	La aplicación ha permitido introducir el objeto en el sistema sin avisar del error.
Cambios realizados	Se ha añadido un pequeño fragmento de código que controla que el nombre del objeto sea único para el sistema para evitar que se produzca este error.

Prueba numero 14

Aplicación a probar	Interfaz (Pestaña NPCS)
Descripción de la prueba	Se va a intentar introducir en el sistema un NPC sin haber elegido una imagen para dicho NPC.
Resultado esperado	Se espera que la aplicación no introduzca el NPC en el sistema e informe al usuario de que es necesario seleccionar una imagen para el NPC
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 15

Aplicación a probar	Interfaz (Pestaña NPCS)
Descripción de la prueba	Se va a intentar introducir en el sistema un NPC sin haber especificado el espacio en el que se encontrara dicho NPC
Resultado esperado	Se espera que la aplicación no introduzca el NPC en el sistema e informe al usuario de que es necesario especificar el espacio donde se encuentra el NPC.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 16

Aplicación a probar	Interfaz (Pestaña NPCS)
Descripción de la prueba	Se va a intentar introducir en el sistema un NPC sin haber especificado las coordenadas del espacio en las que está situado el NPC.
Resultado esperado	Se espera que la aplicación no introduzca el NPC en el sistema e informe al usuario de que es necesario introducir unas coordenadas para dicho NPC.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 17

Aplicación a probar	Interfaz (Pestaña NPCS)
Descripción de la prueba	Se va a intentar introducir en el sistema un NPC en unas coordenadas incorrectas
Resultado esperado	Se espera que la aplicación no introduzca el NPC en el sistema e informe al usuario de que es necesario introducir unas coordenadas para dicho NPC.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 18

Aplicación a probar	Interfaz (Pestaña NPCS)
Descripción de la prueba	Se va a intentar introducir en el sistema un NPC sin haber especificado la acción que realizara al interactuar con el usuario
Resultado esperado	Se espera que la aplicación no introduzca el NPC en el sistema e informe al usuario de que es necesario introducir una acción para dicho NPC.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 19

Aplicación a probar	Interfaz (Pestaña NPCS)
Descripción de la prueba	Se va a intentar introducir en el sistema un NPC eligiendo como acción a realizar una acción de unión y sin especificar al id de la unión sobre la que se realizara la acción.
Resultado esperado	Se espera que la aplicación no introduzca el NPC en el sistema e informe al usuario de que es necesario introducir la id de la unión
Resultado obtenido	La aplicación ha permitido introducir el NPC lo que puede provocar errores al ejecutarse el juego.
Cambios realizados	Se ha añadido un código que controla que se seleccione un identificador de unión en el caso de que se seleccione una acción de unión.

Prueba numero 20

Aplicación a probar	Interfaz (Pestaña NPCS)
Descripción de la prueba	Se va a intentar introducir en el sistema un NPC eligiendo como acción a realizar una acción de espacio y sin especificar el id del espacio sobre el que se realizara la acción.
Resultado esperado	Se espera que la aplicación no introduzca el NPC en el sistema e informe al usuario de que es necesario introducir el id del espacio.
Resultado obtenido	La aplicación ha permitido introducir el NPC lo que puede provocar errores al ejecutarse el juego.
Cambios realizados	Se ha añadido un código que controla que se seleccione un identificador de espacio en el caso de que se seleccione una acción de espacio.

Prueba numero 21

Aplicación a probar	Interfaz (Pestaña NPCS)
Descripción de la prueba	Se va a intentar introducir en el sistema un NPC eligiendo como acción a realizar una acción de objeto y sin especificar el id del objeto sobre el que se realizara la acción.
Resultado esperado	Se espera que la aplicación no introduzca el NPC en el sistema e informe al usuario de que es necesario introducir el id del objeto.
Resultado obtenido	La aplicación ha permitido introducir el NPC lo que puede provocar errores al ejecutarse el juego.
Cambios realizados	Se ha añadido un código que controla que se seleccione un identificador de objeto en el caso de que se seleccione una acción de objeto.

Prueba numero 22

Aplicación a probar	Interfaz (Pestaña NPCS)
Descripción de la prueba	Se va a intentar introducir en el sistema un NPC con el mismo nombre que otro NPC introducido previamente en el sistema
Resultado esperado	Se espera que la aplicación no introduzca el NPC en el sistema e informe al usuario de que el nombre del NPC debe de ser único.
Resultado obtenido	La aplicación ha permitido introducir el NPC en el sistema sin avisar del error.
Cambios realizados	Se ha añadido un pequeño fragmento de código que controla que el nombre del NPC sea único para el sistema para evitar que se produzca este error.

Prueba numero 23

Aplicación a probar	Interfaz (Modulo Conversaciones)
Descripción de la prueba	Se va a intentar crear una nueva conversación sin especificar el número de pasos de la conversación.
Resultado esperado	Se espera que la aplicación no permita pasar al siguiente paso del proceso de creación de conversaciones y que informe al usuario de que es necesario especificar el número de pasos de la conversación.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 24

Aplicación a probar	Interfaz (Modulo Conversaciones)
Descripción de la prueba	Se va a intentar crear una nueva conversación con un nombre que ya se está utilizando para otra conversación introducida previamente en el sistema.
Resultado esperado	Se espera que la aplicación no permita pasar al siguiente paso del proceso de creación de conversaciones y que informe al usuario de que el nombre de la conversación debe de ser único.
Resultado obtenido	La aplicación ha permitido pasar al siguiente paso de creación de la conversación sin avisar del error.
Cambios realizados	Se ha añadido un pequeño fragmento de código que controla que el nombre de la conversación sea único para el sistema para evitar que se produzca este error.

Prueba numero 25

Aplicación a probar	Interfaz (Modulo Conversaciones)
Descripción de la prueba	Se va a elegir una acción sobre un espacio en un paso de la conversación sin especificar la id del espacio sobre el que se realizara la acción.
Resultado esperado	Se espera que la aplicación no permita crear la conversación y que informe al usuario de que es necesario especificar la id del espacio sobre el que se realizara la acción.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 26

Aplicación a probar	Interfaz (Modulo Conversaciones)
Descripción de la prueba	Se va a elegir una acción sobre un objeto en un paso de la conversación sin especificar la id del objeto sobre el que se realizara la acción.
Resultado esperado	Se espera que la aplicación no permita crear la conversación y que informe al usuario de que es necesario especificar la id del objeto sobre el que se realizara la acción.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 27

Aplicación a probar	Interfaz (Modulo Conversaciones)
Descripción de la prueba	Se va a elegir una acción sobre una unión en un paso de la conversación sin especificar la id de la unión sobre la que se realizara la acción.
Resultado esperado	Se espera que la aplicación no permita crear la conversación y que informe al usuario de que es necesario especificar la id de la unión sobre el que se realizara la acción.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 28

Aplicación a probar	Interfaz (Modulo Reglas)
Descripción de la prueba	Se va a intentar crear una nueva regla sin seleccionar ninguna condición para que se ejecute la regla.
Resultado esperado	Se espera que la aplicación no permita crear la regla y que avise al usuario sobre la necesidad de especificar una condición para la regla.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 29

Aplicación a probar	Interfaz (Modulo Reglas)
Descripción de la prueba	Se va a intentar crear una nueva regla sin seleccionar ninguna consecuencia para la regla
Resultado esperado	Se espera que la aplicación no permita crear la regla y que avise al usuario sobre la necesidad de especificar una consecuencia para la regla.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 30

Aplicación a probar	Interfaz (Modulo Reglas)
Descripción de la prueba	Se va a intentar crear una nueva regla con una condición que utilice una id de espacio sin especificar la id del espacio
Resultado esperado	Se espera que la aplicación no permita crear la regla y que avise al usuario sobre la necesidad de especificar la id del espacio.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 31

Aplicación a probar	Interfaz (Modulo Reglas)
Descripción de la prueba	Se va a intentar crear una nueva regla con una consecuencia que utilice una id de unión sin especificar la id de la unión.
Resultado esperado	Se espera que la aplicación no permita crear la regla y que avise al usuario sobre la necesidad de especificar la id de la unión.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 32

Aplicación a probar	Interfaz (Modulo Reglas)
Descripción de la prueba	Se va a intentar crear una nueva regla con una consecuencia que utilice una id de espacio sin especificar la id del espacio.
Resultado esperado	Se espera que la aplicación no permita crear la regla y que avise al usuario sobre la necesidad de especificar la id del espacio.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 33

Aplicación a probar	Interfaz (Modulo Reglas)
Descripción de la prueba	Se va a intentar crear una nueva regla con una consecuencia que utilice una id de objeto sin especificar la id del objeto.
Resultado esperado	Se espera que la aplicación no permita crear la regla y que avise al usuario sobre la necesidad de especificar la id del objeto.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 34

Aplicación a probar	Interfaz (Modulo Reglas)
Descripción de la prueba	Se va a intentar crear una nueva regla con una consecuencia de nueva acción para un NPC, dicha acción será una acción de objeto y no se especificara la id del objeto.
Resultado esperado	Se espera que la aplicación no permita crear la regla y que avise al usuario sobre la necesidad de especificar la id del objeto.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 35

Aplicación a probar	Interfaz (Modulo Reglas)
Descripción de la prueba	Se va a intentar crear una nueva regla con una consecuencia de nueva acción para un NPC, dicha acción será una acción de espacio y no se especificara la id del espacio.
Resultado esperado	Se espera que la aplicación no permita crear la regla y que avise al usuario sobre la necesidad de especificar la id del espacio.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 36

Aplicación a probar	Interfaz (Modulo Reglas)
Descripción de la prueba	Se va a intentar crear una nueva regla con una consecuencia de nueva acción para un NPC, dicha acción será una acción de unión y no se especificara la id de la unión.
Resultado esperado	Se espera que la aplicación no permita crear la regla y que avise al usuario sobre la necesidad de especificar la id del espacio.
Resultado obtenido	El resultado obtenido ha sido el esperado.
Cambios realizados	No ha sido necesario realizar ningún cambio.

Prueba numero 37

Aplicación a probar	Interfaz (Modulo Copiar)
Descripción de la prueba	Se va a intentar crear el instalador del juego sin especificar el directorio donde se quiere crear dicho instalador.
Resultado esperado	Se espera que la aplicación no permita crear el instalador y que avise al usuario sobre la necesidad de especificar un directorio.
Resultado obtenido	Se ha producido un erro en tiempo de ejecución y se ha detenido la ejecución de la aplicación.
Cambios realizados	Se ha controlado el error para evitar que vuelva a ocurrir.

Prueba numero 38

Aplicación a probar	Interfaz (Modulo Instalar)
Descripción de la prueba	Se va a intentar instalar un juego sin especificar el directorio donde se encuentra el instalador de dicho juego.
Resultado esperado	Se espera que la aplicación no permita instalar el juego y que avise al usuario sobre la necesidad de especificar el directorio en el que se encuentra el instalador.
Resultado obtenido	Se ha producido un erro en tiempo de ejecución y se ha detenido la ejecución de la aplicación.
Cambios realizados	Se ha controlado el error para evitar que vuelva a ocurrir.

Prueba numero 39

Aplicación a probar	Interfaz (Modulo Instalar)
Descripción de la prueba	Se va a intentar instalar un juego sin especificar el directorio donde se quiere instalar el juego
Resultado esperado	Se espera que la aplicación no permita instalar el juego y que avise al usuario sobre la necesidad de especificar el directorio en el que se quiere instalar el juego.
Resultado obtenido	El resultado obtenido ha sido el esperado.

Prueba numero 40

Aplicación a probar	Interfaz (Modulo Instalar)
Descripción de la prueba	Se va a intentar instalar un juego especificando como directorio donde se encuentra el instalador un directorio en el que no está el instalador.
Resultado esperado	Se espera que la aplicación no permita instalar el juego y que avise al usuario de que no se ha podido encontrar el instalador en la ruta especificada.
Resultado obtenido	El resultado obtenido ha sido el esperado.

Prueba numero 41

Aplicación a probar	Interfaz (Modulo Instalar)
Descripción de la prueba	Se va a intentar instalar un juego en el que se haya perdido el archivo de configuración config.txt.
Resultado esperado	Se espera que la aplicación no permita instalar el juego y que avise al usuario de que se ha perdido uno de los archivos de configuración del juego.
Resultado obtenido	El resultado obtenido ha sido el esperado.

Prueba numero 42

Aplicación a probar	Interfaz (Modulo Instalar)
Descripción de la prueba	Se va a intentar instalar un juego en el que se haya perdido el archivo de configuración conversaciones.txt.
Resultado esperado	Se espera que la aplicación no permita instalar el juego y que avise al usuario de que se ha perdido uno de los archivos de configuración del juego.
Resultado obtenido	El resultado obtenido ha sido el esperado.

Prueba numero 43

Aplicación a probar	Interfaz (Modulo Instalar)
Descripción de la prueba	Se va a intentar instalar un juego en el que se haya perdido el archivo de configuración espacios.txt.
Resultado esperado	Se espera que la aplicación no permita instalar el juego y que avise al usuario de que se ha perdido uno de los archivos de configuración del juego.
Resultado obtenido	El resultado obtenido ha sido el esperado.

Prueba numero 44

Aplicación a probar	Interfaz (Modulo Instalar)
Descripción de la prueba	Se va a intentar instalar un juego en el que se haya perdido el archivo de configuración fondos.txt.
Resultado esperado	Se espera que la aplicación no permita instalar el juego y que avise al usuario de que se ha perdido uno de los archivos de configuración del juego.
Resultado obtenido	El resultado obtenido ha sido el esperado.

Prueba numero 45

Aplicación a probar	Interfaz (Modulo Instalar)
Descripción de la prueba	Se va a intentar instalar un juego en el que se haya perdido el archivo de configuración npcs.txt.
Resultado esperado	Se espera que la aplicación no permita instalar el juego y que avise al usuario de que se ha perdido uno de los archivos de configuración del juego.
Resultado obtenido	El resultado obtenido ha sido el esperado.

Prueba numero 46

Aplicación a probar	Interfaz (Modulo Instalar)
Descripción de la prueba	Se va a intentar instalar un juego en el que se haya perdido el archivo de configuración reglas.txt.
Resultado esperado	Se espera que la aplicación no permita instalar el juego y que avise al usuario de que se ha perdido uno de los archivos de configuración del juego.
Resultado obtenido	El resultado obtenido ha sido el esperado.

Prueba numero 47

Aplicación a probar	Interfaz (Modulo Instalar)
Descripción de la prueba	Se va a intentar instalar un juego en el que se haya perdido el archivo de configuración uniones.txt.
Resultado esperado	Se espera que la aplicación no permita instalar el juego y que avise al usuario de que se ha perdido uno de los archivos de configuración del juego.
Resultado obtenido	El resultado obtenido ha sido el esperado.

Prueba numero 48

Aplicación a probar	Interfaz (Modulo Instalar)
Descripción de la prueba	Se va a intentar instalar un juego en el que se haya perdido el archivo del juego Game.exe
Resultado esperado	Se espera que la aplicación no permita instalar el juego y que avise al usuario de que se ha perdido el archivo ejecutable del juego.
Resultado obtenido	El resultado obtenido ha sido el esperado.

Prueba numero 49

Aplicación a probar	Interfaz (Modulo Instalar)
Descripción de la prueba	Se va a intentar instalar un juego en el que se hayan perdido la carpeta Content del videojuego.
Resultado esperado	Se espera que la aplicación no permita instalar el juego y que avise al usuario de que se han perdido archivos necesarios para la ejecución del juego.
Resultado obtenido	El resultado obtenido ha sido el esperado.

Prueba numero 50

Aplicación a probar	Compilador de imágenes
Descripción de la prueba	Se va a intentar insertar una nueva imagen para objeto con un nombre que ya se está utilizando para otra imagen introducida previamente en el sistema.
Resultado esperado	Se espera que la aplicación no permita introducir la imagen en el sistema y avise el usuario de que el nombre de la imagen debe de ser único
Resultado obtenido	El resultado obtenido ha sido el esperado.

Prueba numero 51

Aplicación a probar	Compilador de imágenes
Descripción de la prueba	Se va a intentar insertar una nueva imagen para fondo con un nombre que ya se está utilizando para otra imagen introducida previamente en el sistema.
Resultado esperado	Se espera que la aplicación no permita introducir la imagen en el sistema y avise el usuario de que el nombre de la imagen debe de ser único
Resultado obtenido	El resultado obtenido ha sido el esperado.